Graphical Models in Machine Learning

AI4190

Outlines of Tutorial

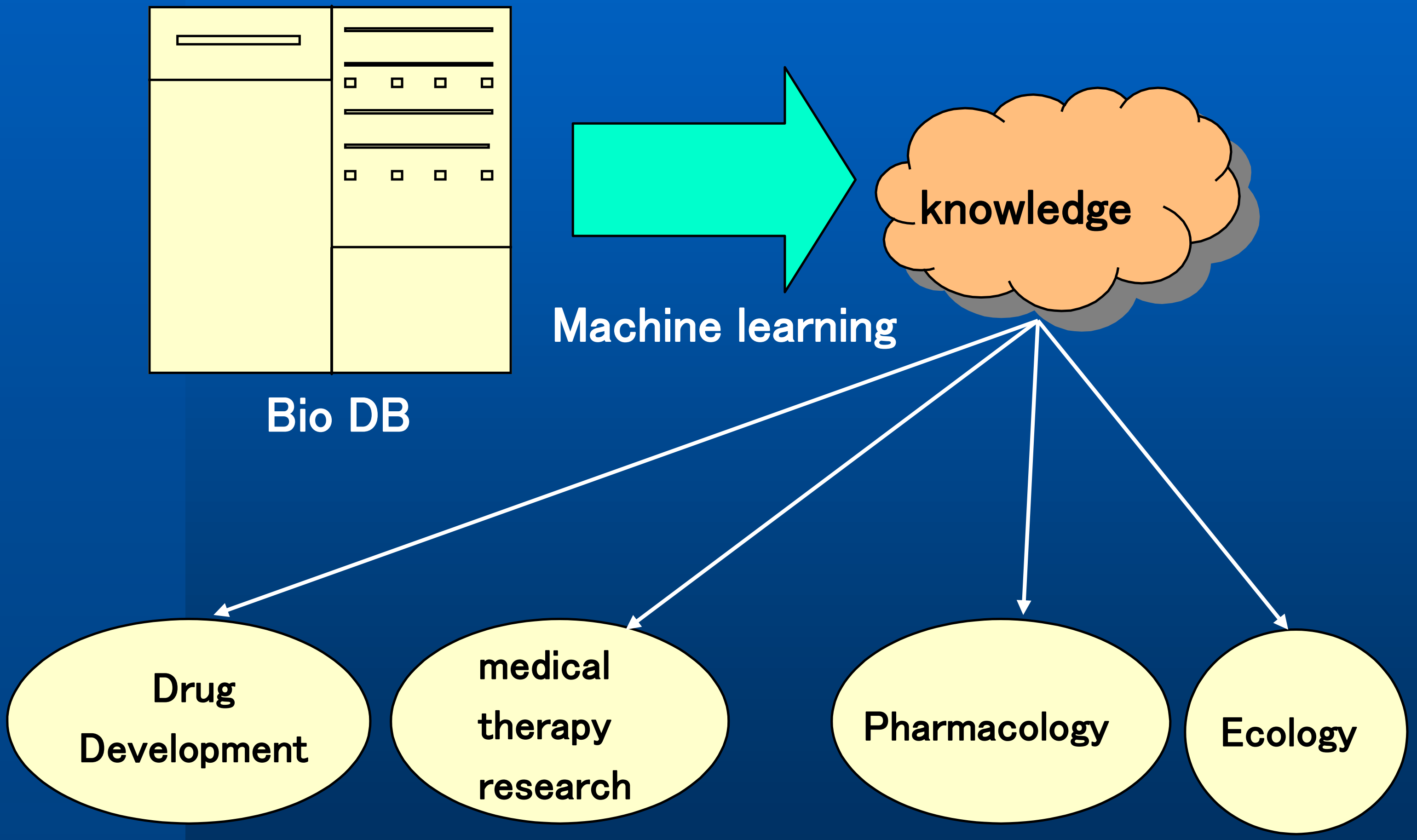
1. Machine Learning and Bioinformatics
   - Machine Learning
   - Problems in Bioinformatics
   - Machine Learning Methods
   - Applications of ML Methods for Bio Data Mining
2. Graphical Models
   - Bayesian Network
   - Generative Topographic Mapping
   - Probabilistic clustering
   - NMF (nonnegative matrix factorization)

Outlines of Tutorial

3. Other Machine Learning Methods
   - Neural Networks
   - K Nearest Neighborhood
   - Radial Basis Function
4. DNA Microarrays
5. Applications of GTM for Bio Data Mining
   - DNA Chip Gene Expression Data Analysis
   - Clustering the Genes
6. Summary and Discussion

* References

1. Machine Learning and Bioinformatics

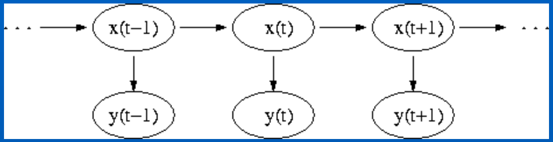


Machine Learning

- Supervised Learning
  - Estimate an unknown mapping from known input- output pairs
  - Learn $f_{\mathbf{w}}$ from training set $D=\{(\mathbf{x},y)\}$ s.t. $f_{\mathbf{w}}(\mathbf{x}) = y = f(\mathbf{x})$
  - Classification: y is discrete, categorical
  - Regression: y is continuous
- Unsupervised Learning
  - Only input values are provided
  - Learn $f_{\mathbf{w}}$ from $D=\{(\mathbf{x})\}$ $f_{\mathbf{w}}(\mathbf{x}) = y$
  - Compression
  - Clustering

Machine Learning Methods

- Probabilistic Models
  - Hidden Markov Models
  - Bayesian Networks
  - Generative Topographic Mapping (GTM)
- Neural Networks
  - Multilayer Perceptrons (MLPs)
  - Self-Organizing Maps (SOM)
- Genetic Algorithms
- Other Machine Learning Algorithms
  - Support Vector Machines
  - Nearest Neighbor Algorithms
  - Decision Trees



Applications of ML Methods for Bio Data Mining (1)

- Structure and Function Prediction
  - Hidden Markov Models
  - Multilayer Perceptrons
  - Decision Trees
- Molecular Clustering and Classification
  - Support Vector Machines
  - Nearest Neighbor Algorithms
- Expression (DNA Chip Data) Analysis:
  - Self-Organizing Maps
  - Bayesian Networks
  - Generative Topographic Mapping
- Bayesian Networks
  - Gene Modeling → Gene Expression Analysis
  - [Friedman et al., 2000]

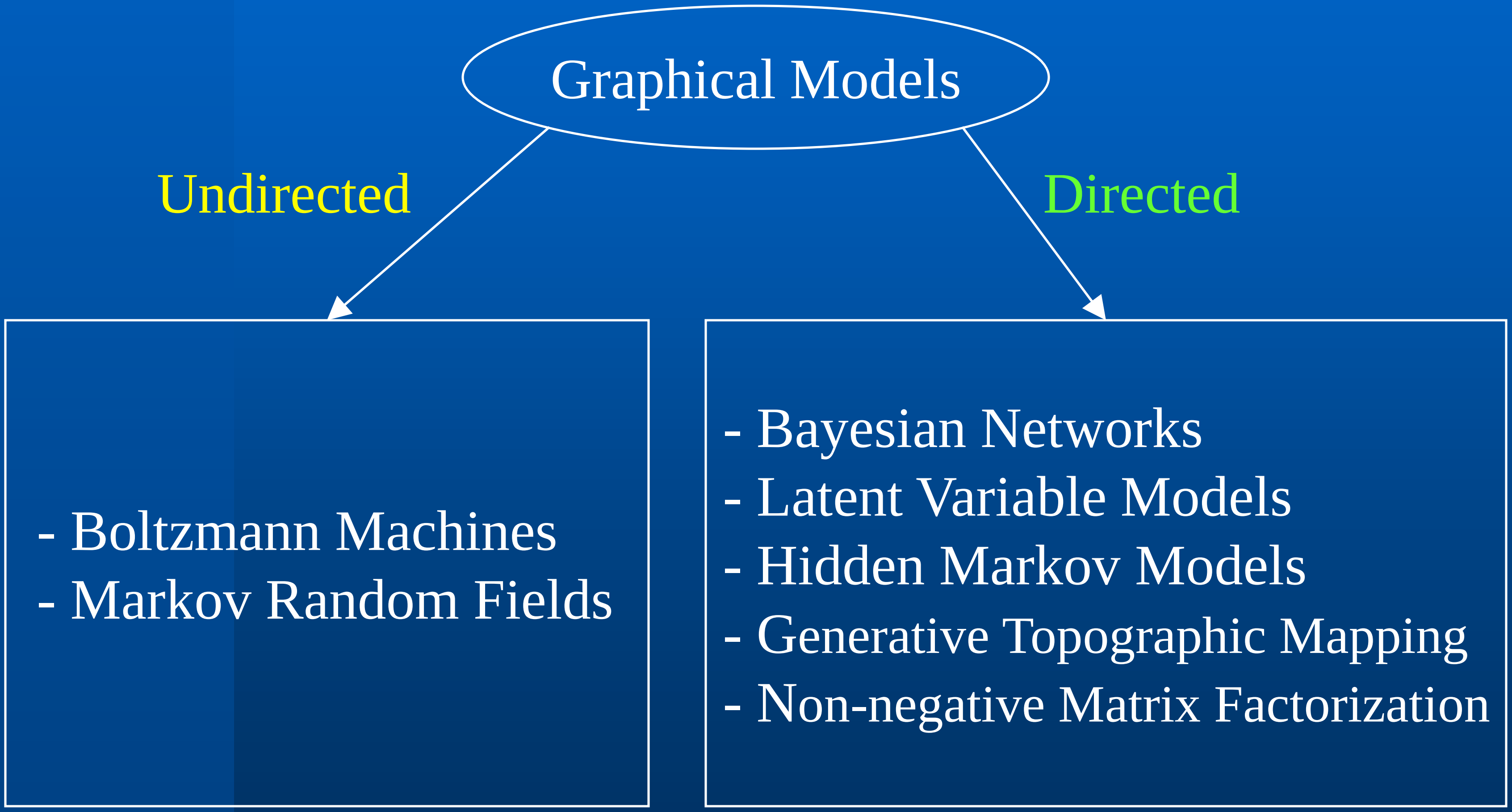
Applications of ML Methods for Bio Data Mining (2)

- Multi-layer Perceptrons
  - Gene Finding / Structure Prediction
  - Protein Modeling / Structure and Function Prediction
- Self-Organizing Maps (Kohonen Neural Network)
  - Molecular Clustering
  - DNA Chip Gene Expression Data Analysis
- Support Vector Machines
  - Classification of Microarray Gene Expression and Gene Functional Class
- Nearest Neighbor Algorithms
  - 3D Protein Classification
- Decision Trees
  - Gene Finding: MORGAN system
  - Molecular Clustering

2. Probabilistic Graphical Models

- Represent the joint probability distribution on some random variables in compact form.
  - Undirected probabilistic graphical models
    - Markov random fields
    - Boltzmann machines
  - Directed probabilistic graphical models
    - Helmholtz machines
    - Bayesian networks
- Probability distribution for some variables given values of other variables can be obtained in a probabilistic graphical model.
  - Probabilistic inference.

Classes of Graphical Models



Graphical Models

Undirected

Directed

- Boltzmann Machines
- Markov Random Fields

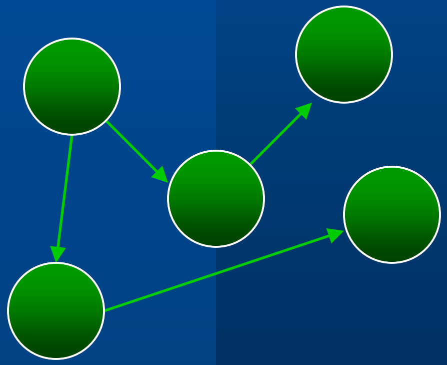
- Bayesian Networks
- Latent Variable Models
- Hidden Markov Models
- Generative Topographic Mapping
- Non-negative Matrix Factorization

- Bayesian Networks

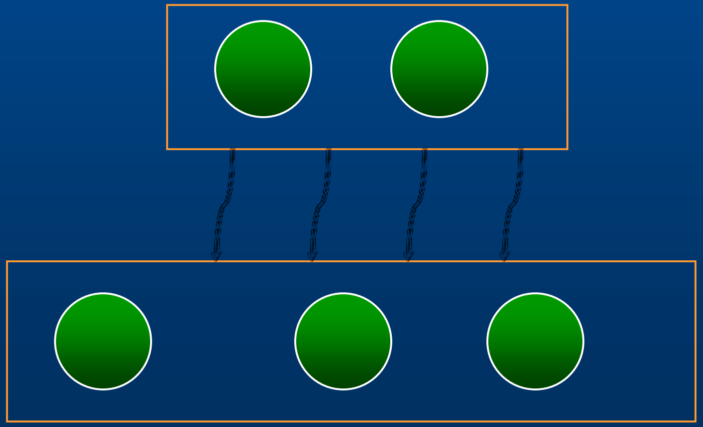
  A graphical model for probabilistic relationships among a set of variables

- Generative Topographic Mapping

  A graphical model through a nonlinear relationship between the latent variables and observed features.



(Bayesian Network)



(GTM)

Bayesian Networks

Contents

- Introduction
- Bayesian approach
- Bayesian networks
- Inferences in BN
- Parameter and structure learning
- Search methods for network
- Case studies
- Reference

Introduction

- Bayesian network is a graphical network for expressing the dependency relations between features or variables
- BN can learn the casual relationships for the understanding of the problem domain
- BN offers an efficient way of avoiding the over fitting of the data (model averaging, model selection)
- Scores for network structure fitness: BDe, MDL, BIC

Bayesian approach

- Bayesian probability: a person's degree of belief
- Thumbtack example: After N flips, probability of heads on the (N+1)th toss = ?
  - Classic analysis: estimate this probability from the N observations with low variance and bias

    $$E(\theta^*) = \sum_D p(D \mid \theta)\theta^*(D)$$

    $$Var(\theta^*) = \sum_D p(D \mid \theta)(\theta^*(D) - E(\theta^*))^2$$

  - Ex) ML estimator: choose θ to maximize the likelihood $p(D \mid \theta)$
  - Bayesian approach: D is fixed and imagine all the possible θ from this D

    $$E(\theta) = \int \theta p(\theta \mid D, h) d\theta$$

Bayesian approach

- Bayesian approach:

$$\overset{\text{posterior}}{\boxed{p(\theta\,|\,D,\xi)}}=\frac{\overset{\text{prior}}{\boxed{p(\theta\,|\,\xi)}}\,\overset{\text{likelihood}}{\boxed{p(D\,|\,\theta,\xi)}}}{p(D\,|\,\xi)}=\frac{p(\theta\,|\,\xi)\;\theta^{h}(1-\theta)^{t}}{p(D\,|\,\xi)}$$

$$p(D\,|\,\xi)=\int p(D\,|\,\theta,\xi)\,p(\theta\,|\,\xi)\,d\theta\quad\text{(marginal likelihood)}$$

$$\underline{p(X_{N+1}=heads\,|\,D,\xi)=\int p(X_{N+1}=heads\,|\,\theta,\xi)p(\theta\,|\,D,\xi)d\theta}$$

$$=\int\theta\,p(\theta\,|\,D,\xi)d\theta=E(\theta)$$

- Conjugate prior has posterior as the same family of distribution w.r.t. the likelihood distribution
  - Normal likelihood - Normal prior - Normal posterior
  - Binomial likelihood - Beta prior - Beta posterior
  - Multinomial likelihood - Dirichlet prior- Dirichlet posterior
  - Poisson likelihood - Gamma prior - Gamma posterior

$$p(x,\theta\,|\,D)=p(x\,|\,\theta,D)\,p(\theta\,|\,D)=p(x\,|\,\theta)\,p(\theta\,|\,D)$$

Bayesian approach

$$p(X = Head \mid \theta, \xi) = \theta,$$

$$p(\theta \mid \xi) = Beta(\theta \mid \alpha_h, \alpha_t) \equiv \frac{\Gamma(\alpha)}{\Gamma(\alpha_h)\Gamma(\alpha_t)} \theta^{\alpha_h - 1}(1-\theta)^{\alpha_t - 1}, \quad (\alpha = \alpha_h + \alpha_t)$$

$$p(\theta \mid D, \xi) = \frac{\Gamma(\alpha + N)}{\Gamma(\alpha_h + h)\Gamma(\alpha_t + t)} \theta^{\alpha_h + h - 1}(1-\theta)^{\alpha_t + t - 1} = Beta(\theta \mid \alpha_h + h, \alpha_t + t)$$

$$\int \theta \; Beta(\theta \mid \alpha_h, \alpha_t) d\theta = \frac{\alpha_h}{\alpha}$$

$$p(X_{N+1} = heads \mid D, \xi) = \int p(X_{N+1} = heads \mid \theta, \xi) p(\theta \mid D, \xi) d\theta = \frac{\alpha_h + h}{\alpha + N}$$

$$p(X = x^k \mid \boldsymbol{\theta}, \xi) = \theta_k, \quad k = 1, \cdots, r$$

$$p(\boldsymbol{\theta} \mid \xi) = Dir(\boldsymbol{\theta} \mid \alpha_1, \cdots, \alpha_r) \equiv \frac{\Gamma(\alpha)}{\prod_{k=1}^{r} \Gamma(\alpha_k)} \prod_{k=1}^{r} \theta_k^{\alpha_k - 1}, \; (\alpha = \textstyle\sum_{\mathrm{k=1}}^{\mathrm{r}} \alpha_{\mathrm{k}}) \; \text{(prior)}$$

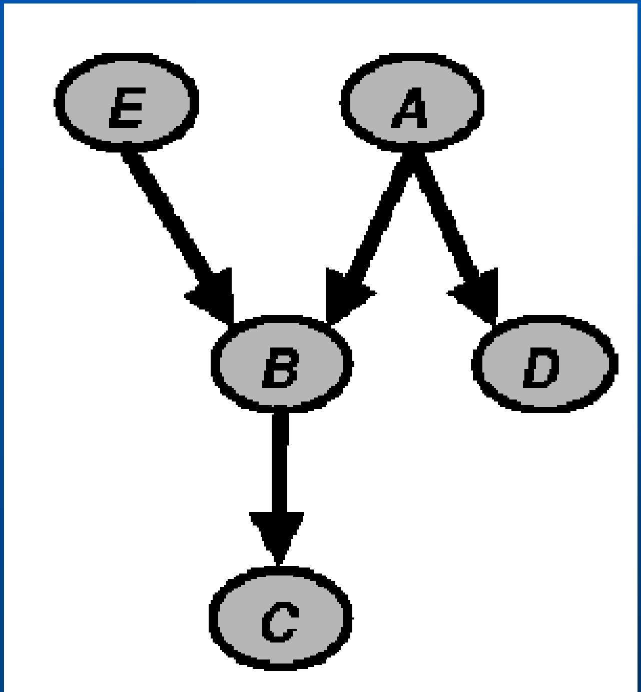
$$p(\boldsymbol{\theta} \mid D, \xi) = Dir(\boldsymbol{\theta} \mid \alpha_1 + N_1, \cdots, \alpha_r + N_r) \quad \text{(posterior)}$$

$$p(X_{N+1} = x^k \mid D, \xi) = \int \theta_k Dir(\boldsymbol{\theta} \mid \alpha_1 + N_1, \cdots, \alpha_r + N_r) d\boldsymbol{\theta} = \frac{\alpha_k + N_k}{\alpha + N}$$

$$p(D \mid \xi) = \frac{\Gamma(\alpha)}{\Gamma(\alpha + N)} \prod_{k=1}^{r} \frac{\Gamma(\alpha_k + N_k)}{\Gamma(\alpha_k)} \quad \text{(marginal likelihood or evidence)}$$

Bayesian Networks (1) -Architecture

- Bayesian networks represent statistical relationships among random variables (e.g. genes).



- *B* and *D* are independent given *A*.

- *B* asserts dependency between *A* and *E*.

- *A* and *C* are independent given *B*.

$$P(A,B,C,D,E)$$
$$= P(A)P(B \mid A,E)P(C \mid B)P(D \mid A)P(E)$$

Bayesian Networks (1) -example

$$P(X_1,X_2,X_3)=P(X_1\,|\,X_2,X_3)P(X_2,X_3)$$
$$=P(X_1\,|\,X_2,X_3)P(X_2\,|\,X_3)P(X_3)$$

- BN = (S, P) consists a network structure S and a set of local probability distributions P $\quad p(\mathbf{x})=\prod_{i=1}^{n}p(x_i\,|\,\mathbf{pa}_i)$

$$p(\mathbf{x})=\prod_{i=1}^{n}p(x_i\,|\,x_1,\cdots,x_{i-1})\quad \text{(chain rule)}$$

$$p(x_i\,|\,x_1,\cdots,x_{i-1})\cong p(x_i\,|\,\pi_i)$$

$$p(\mathbf{x})=\prod_{i=1}^{n}p(x_i\,|\,\pi_i)$$

(1) order variables : (F, A, S, G, J),

note : search of $n!$ cases in the worst case

(2) find π_i :

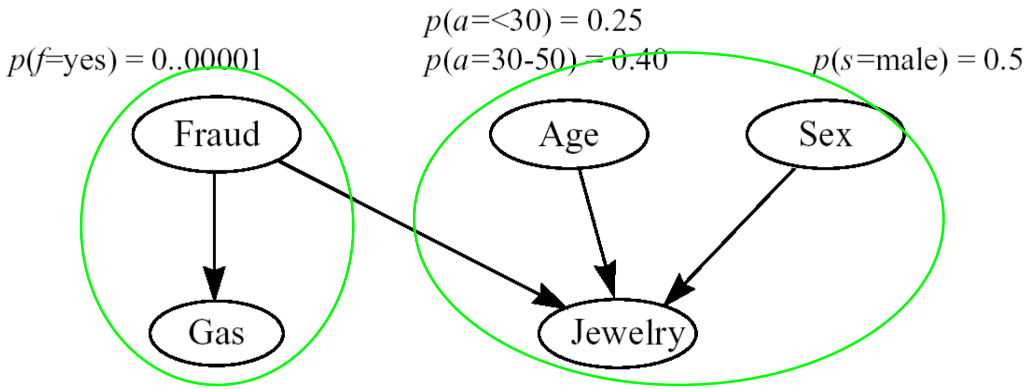
$$p(a\,|\,f)=p(a)$$

$$p(s\,|\,f,a)=p(s)$$

$$p(g\,|\,f,a,s)=p(g\,|\,f)$$

$$p(j\,|\,f,a,s,g)=p(j\,|\,f,a,s)$$

<BN for detecting credit card fraud>



$p(f{=}\text{yes}) = 0..00001$

$p(a{=}{<}30) = 0.25$
$p(a{=}30\text{-}50) = 0.40$

$p(s{=}\text{male}) = 0.5$

$p(g{=}yes|f{=}yes) = 0.2$
$p(g{=}yes|f{=}no) = 0.01$

$p(j{=}yes|f{=}yes,a{=}*,s{=}*) = 0.05$
$p(j{=}yes|f{=}no,a{=}{<}30,s{=}male) = 0..0001$
$p(j{=}yes|f{=}no,a{=}30\text{-}50,s{=}male) = 0.0004$
$p(j{=}yes|f{=}no,a{=}{>}50,s{=}male) = 0.0002$
$p(j{=}yes|f{=}no,a{=}{<}30,s{=}female) = 0..0005$
$p(j{=}yes|f{=}no,a{=}30\text{-}50,s{=}female) = 0.002$
$p(j{=}yes|f{=}no,a{=}{>}50,s{=}female) = 0.001$

Case	Fraud	Gas	Jewelry	Age	Sex
1	no	no	no	30-50	female
2	no	no	no	30-50	male
3	yes	yes	yes	>50	male
4	no	no	no	30-50	male
5	no	yes	no	<30	female
6	no	no	no	<30	female
7	no	no	no	>50	male
8	no	no	yes	30-50	female
9	no	yes	no	<30	male
10	no	no	no	<30	female

- Structure can be found by relying on the prior knowledge of casual relationships

Bayesian Networks (2) -Characteristics

- DAG (Directed Acyclic Graph)
- Bayesian Network: Network Structure (S) + Local Probability (P).
- Express dependence relations between variables
- Can use prior knowledge on the data (parameter)
  - Dirichlet for multinomial data
  - Normal-Wishart for normal data
- Methods of searching:

  Greedy, Reverse, Exhaustive

Bayesian Networks (3)

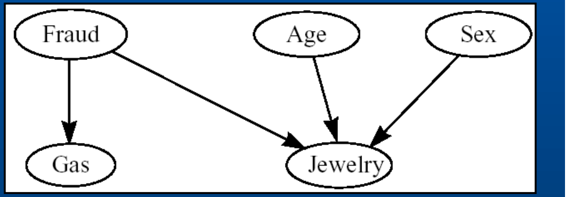
- For missing values:
  - Gibbs sampling
  - Gaussian Approximation
  - EM
  - Bound and Collapse etc.
- Interpretations:
  - Depends on the prior order of nodes or prior structure.
  - Local conditional probability
  - Choice of nodes
  - Overall nature of data

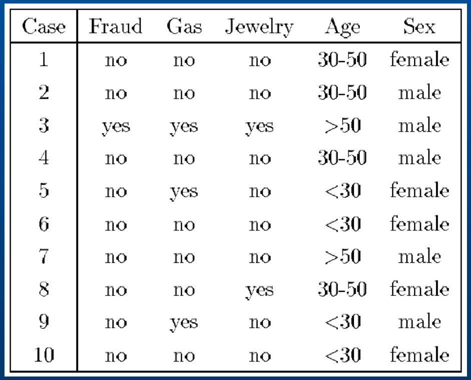
Inferences in BN

$$p(f\,|\,a,s,g,j)=\frac{p(f,a,s,g,j)}{p(a,s,g,j)}=\frac{p(f,a,s,g,j)}{\sum_{f'}p(f',a,s,g,j)}$$

$$p(f\,|\,a,s,g,j)=\frac{p(f)p(a)p(s)p(g\,|\,f)p(j\,|\,f,a,s)}{\sum_{f'}p(f')p(a)p(s)p(g\,|\,f')p(j\,|\,f',a,s)}$$

$$=\frac{p(f)p(g\,|\,f)p(j\,|\,f,a,s)}{\sum_{f'}p(f')p(g\,|\,f')p(j\,|\,f',a,s)}$$





Case	Fraud	Gas	Jewelry	Age	Sex
1	no	no	no	30-50	female
2	no	no	no	30-50	male
3	yes	yes	yes	>50	male
4	no	no	no	30-50	male
5	no	yes	no	<30	female
6	no	no	no	<30	female
7	no	no	no	>50	male
8	no	no	yes	30-50	female
9	no	yes	no	<30	male
10	no	no	no	<30	female

- A tutorial on learning with Bayesian networks (David Heckerman)

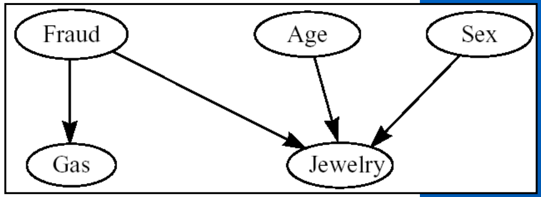
Inferences in BN (parameter learning)

$$p(\mathbf{x}|\boldsymbol{\theta}_S, S^h) = \prod_{i=1}^{n} p(x_i|\mathbf{pa}_i, \boldsymbol{\theta}_i, S^h)$$

$$p(x_i^k|\mathbf{pa}_i^j, \boldsymbol{\theta}_i, S^h) = \theta_{ijk} > 0, \quad \boldsymbol{\theta}_{ij} = (\theta_{ij2}, \cdots, \theta_{ijr_i})$$

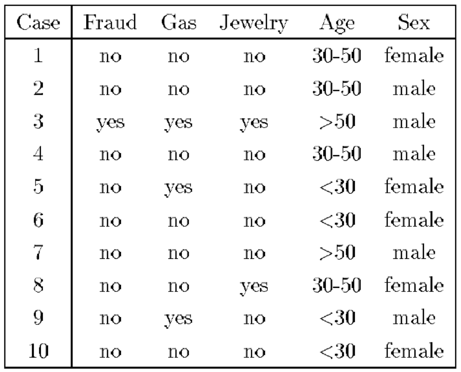
x_i has r_i possible discrete values $x_i^1, \cdots, x_i^{r_i}$ $(k \in \{1, \cdots, r_i\})$

$\mathbf{pa}_i$ has $\prod_{X_i \in Pa_i} r_i = q_i$ discrete combination values (j $\in \{1, \cdots, q_i\}$)



$$[p(X_{N+1} = heads|D, \xi) = \int p(X_{N+1} = heads|\theta, \xi) p(\theta|D, \xi) d\theta = \int \theta p(\theta|D, \xi) d\theta = E(\theta)]$$

$$p(X_{N+1} = x^k|D, \xi) = \int \theta_k Dir(\boldsymbol{\theta}|\alpha_1 + N_1, \cdots, \alpha_r + N_r) d\boldsymbol{\theta} = \frac{\alpha_k + N_k}{\alpha + N}$$



Case	Fraud	Gas	Jewelry	Age	Sex
1	no	no	no	30-50	female
2	no	no	no	30-50	male
3	yes	yes	yes	>50	male
4	no	no	no	30-50	male
5	no	yes	no	<30	female
6	no	no	no	<30	female
7	no	no	no	>50	male
8	no	no	yes	30-50	female
9	no	yes	no	<30	male
10	no	no	no	<30	female

$p(\boldsymbol{\theta}_S|S^h) = \prod_{i=1}^{n}\prod_{j=1}^{q_i} p(\boldsymbol{\theta}_{ij}|S^h)$ (assume $\boldsymbol{\theta}_{ij}$'s are mutually independent)

$$p(\boldsymbol{\theta}_S|D, S^h) = \prod_{i=1}^{n}\prod_{j=1}^{q_i} p(\boldsymbol{\theta}_{ij}|D, S^h)$$

$$p(\boldsymbol{\theta}_{ij}|D, S^h) = Dir(\boldsymbol{\theta}_{ij}|\alpha_{ij1} + N_{ij1}, \cdots, \alpha_{ijr_i} + N_{ijr_i})$$

$$p(\mathbf{x}_{N+1}|D, S^h) = \int \prod_{i=1}^{n} \theta_{ijk} p(\boldsymbol{\theta}_{ij}|D, S^h) d\boldsymbol{\theta}_S = \prod_{i=1}^{n} \int \theta_{ijk} p(\boldsymbol{\theta}_{ij}|D, S^h) d\boldsymbol{\theta}_{ij}$$

$p(\mathbf{x}_{N+1}|D, S^h) = \prod_{i=1}^{n} \frac{\alpha_{ijk} + N_{ijk}}{\alpha_{ij} + N_{ij}}$ (where $\alpha_{ij} = \sum_{k=1}^{r_i} \alpha_{ijk}$, $N_{ij} = \sum_{k=1}^{r_i} N_{ijk}$, j is pre-chosen)

$$p(X_{N+1} = x^k|D, \xi) = \int \theta_k Dir(\theta|\alpha_1 + N_1, \cdots, \alpha_r + N_r) d\theta = \frac{\alpha_k + \mathrm{N}_k}{\alpha + \mathrm{N}}$$

Parameter and structure learning

Predicting the next case:

$$p(\mathbf{x}_{N+1} | D) = \sum_{S^h} p(\mathbf{x}_{N+1}, S^h | D) = \sum_{S^h} p(\mathbf{x}_{N+1} | D, S^h)\, p(S^h | D)$$

$$= \sum_{S^h} \boxed{p(S^h | D)} \int p(\mathbf{x}_{N+1} | \boldsymbol{\theta}_s, S^h)\, \boxed{p(\boldsymbol{\theta}_s | D, S^h)}\, d\boldsymbol{\theta}_s$$

Bde score

posterior

$$p(S^h | D) = p(S^h)\, \underline{p(D | S^h)} / p(D), \quad \text{(BD score)}$$

* marginal likelihood $p(D | S^h)$

$$p(D | \xi) = \frac{\Gamma(\alpha)}{\Gamma(\alpha + N)} \prod_{k=1}^{r} \frac{\Gamma(\alpha_k + N_k)}{\Gamma(\alpha_k)}$$

$$p(D | S^h) = \prod_{i=1}^{n} \prod_{j=1}^{q_i} \frac{\Gamma(\alpha_{ij})}{\Gamma(\alpha_{ij} + N_{ij})} \prod_{k=1}^{r_i} \frac{\Gamma(\alpha_{ijk} + N_{ijk})}{\Gamma(\alpha_{ijk})}$$

- Averaging over possible models: bottleneck in computations
  - Model selection
  - Selective model averaging

Search method for network structure

- Greedy search :
  - First choose a network structure
  - Evaluate Δ(e) for all e ε E and make the change e for which Δ(e) is maximum. (E: set of eligible changes to graph, Δ(e): the change in log score.)
  - Terminate the search when there is no e with positive Δ(e).
- Avoiding local maxima by simulated annealing
  - Initialize the system at some temperature T_0
  - Pick some eligible change e at random and evaluate $p=\exp(\Delta(e)/T_0)$
  - If p>1 make the change; otherwise make the change with probability p.
  - Repeat this process α times or until make β changes
  - If no changes, lower the temperature and continue the process
  - Stop if the temperature is lowered more than δ times

Example

- A database is given and the possible structures are S_1(figure) and S_2(same with an arc added from Age to Gas) for fraud detection problem.

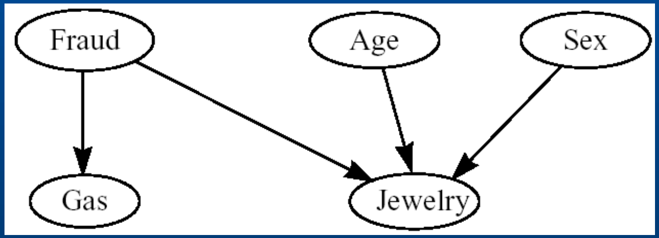
$$p(S^h \mid D) = p(S^h)\,p(D \mid S^h) / p(D), \;\; p(D \mid S^h) = \prod_{i=1}^{n} \prod_{j=1}^{q_i} \frac{\Gamma(\alpha_{ij})}{\Gamma(\alpha_{ij} + N_{ij})} \prod_{k=1}^{r_i} \frac{\Gamma(\alpha_{ijk} + N_{ijk})}{\Gamma(\alpha_{ijk})}$$

$$p(\mathbf{x}_{N+1} \mid D) = \sum_{S^h} p(\mathbf{x}_{N+1}, S^h \mid D) = \sum_{S^h} p(\mathbf{x}_{N+1} \mid D, S^h)\,\underline{p(S^h \mid D)}, \;\; p(\mathbf{x}_{N+1} \mid D, S^h) = \prod_{i=1}^{n} \frac{\alpha_{ijk} + N_{ijk}}{\alpha_{ij} + N_{ij}}$$

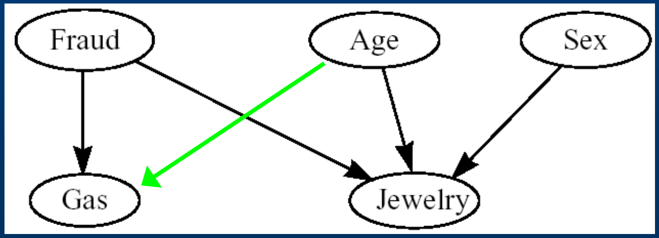
$$p(S_1^h \mid D) = 0.26, \;\; p(S_2^h \mid D) = 0.74,$$

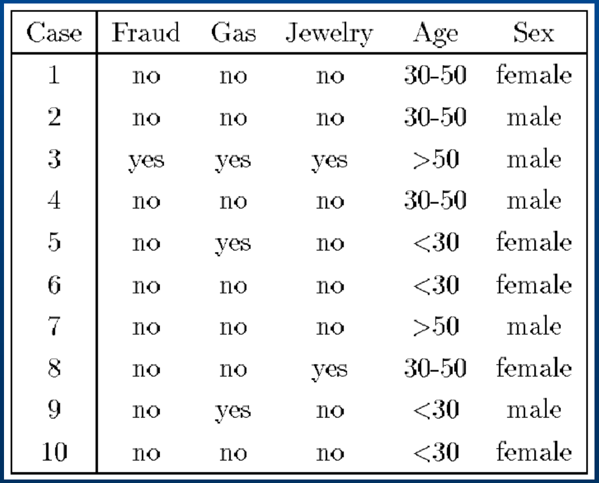
$$p(\mathbf{x}_{N+1} \mid D) = 0.26\,p(\mathbf{x}_{N+1} \mid D, S_1^h) + 0.74\,p(\mathbf{x}_{N+1} \mid D, S_2^h)$$

S_1



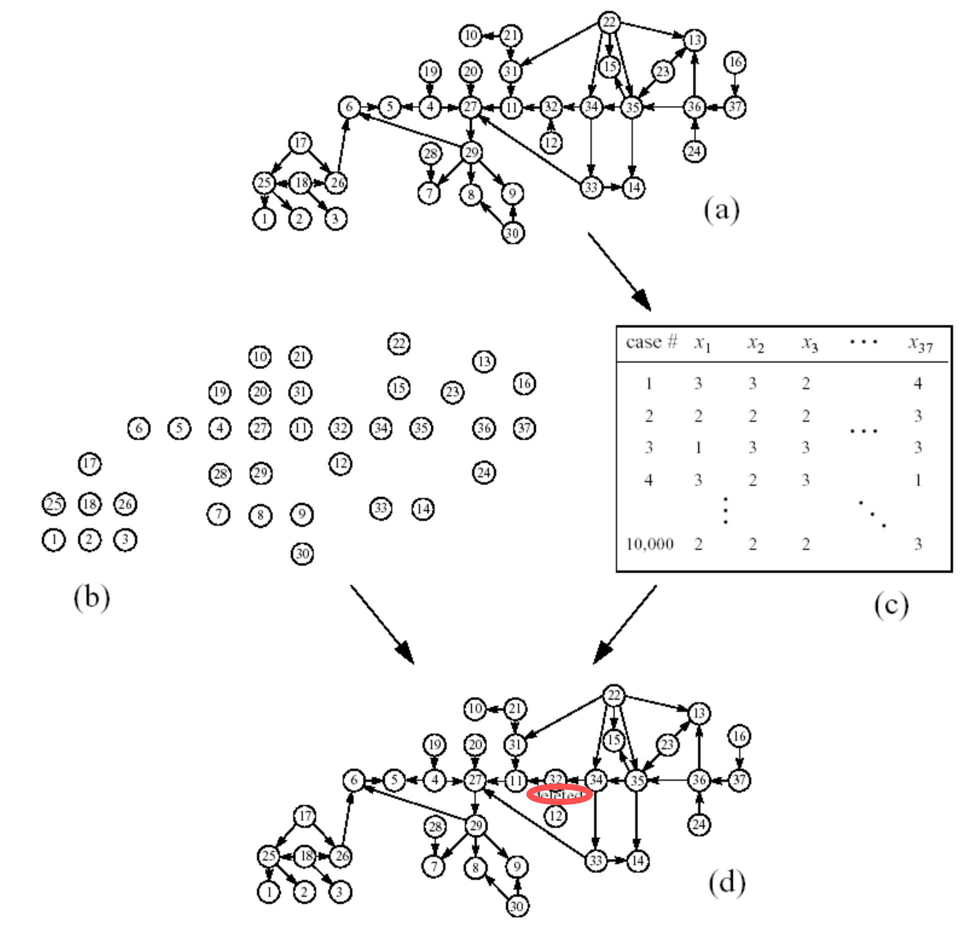
S_2





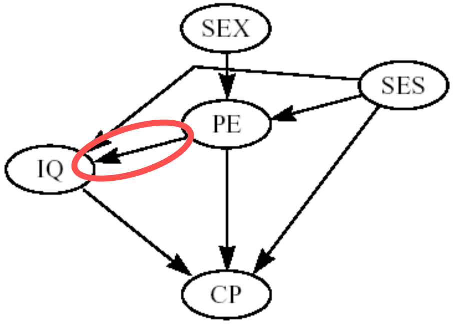
Case	Fraud	Gas	Jewelry	Age	Sex
1	no	no	no	30-50	female
2	no	no	no	30-50	male
3	yes	yes	yes	>50	male
4	no	no	no	30-50	male
5	no	yes	no	<30	female
6	no	no	no	<30	female
7	no	no	no	>50	male
8	no	no	yes	30-50	female
9	no	yes	no	<30	male
10	no	no	no	<30	female

Case studies (1)



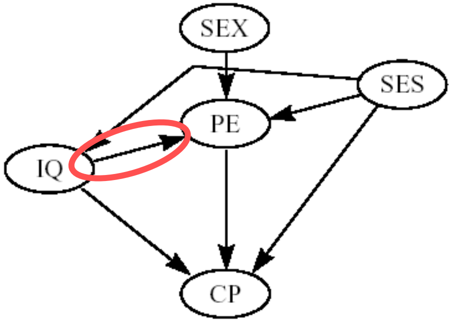
case #	x_1	x_2	x_3	· · ·	x_{37}
1	3	3	2		4
2	2	2	2	· · ·	3
3	1	3	3		3
4	3	2	3		1
	⋮			⋱	
10,000	2	2	2		3

Case studies (2)



$\log p(D|S_1^h) = -45653$

$p(S_1^h|D) = 1.0$



$\log p(D|S_2^h) = -45699$

$p(S_2^h|D) = 1.2 \times 10^{-10}$

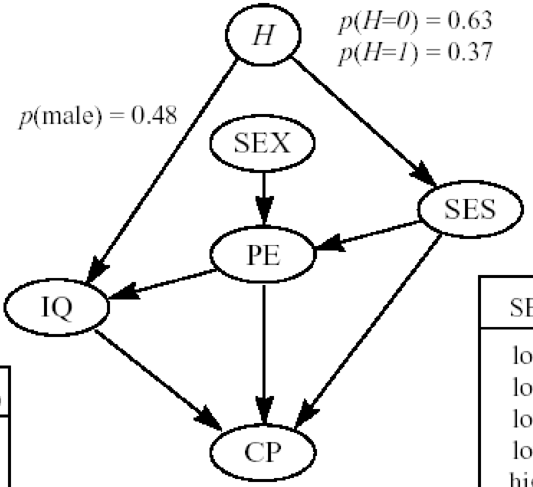
PE: parental encouragement

SES: Socioeconomic status

CP: college plans

PE	H	p(IQ=high|PE,H)
low	0	0.098
low	1	0.22
high	0	0.21
high	1	0.49

SES	SEX	p(PE=high|SES,SEX)
low	male	0.32
low	female	0.166
high	male	0.86
high	female	0.81



$p(male) = 0.48$

$p(H{=}0) = 0.63$
$p(H{=}1) = 0.37$

$\log p(S^h|D) \cong -45629$

H	p(SES=high|H)
low	0.088
high	0.51

SES	IQ	PE	p(CP=yes|SES,IQ,PE)
low	low	low	0.011
low	low	high	0.170
low	high	low	0.124
low	high	high	0.53
high	low	low	0.093
high	low	high	0.39
high	high	low	0.24
high	high	high	0.84

Case studies (3)

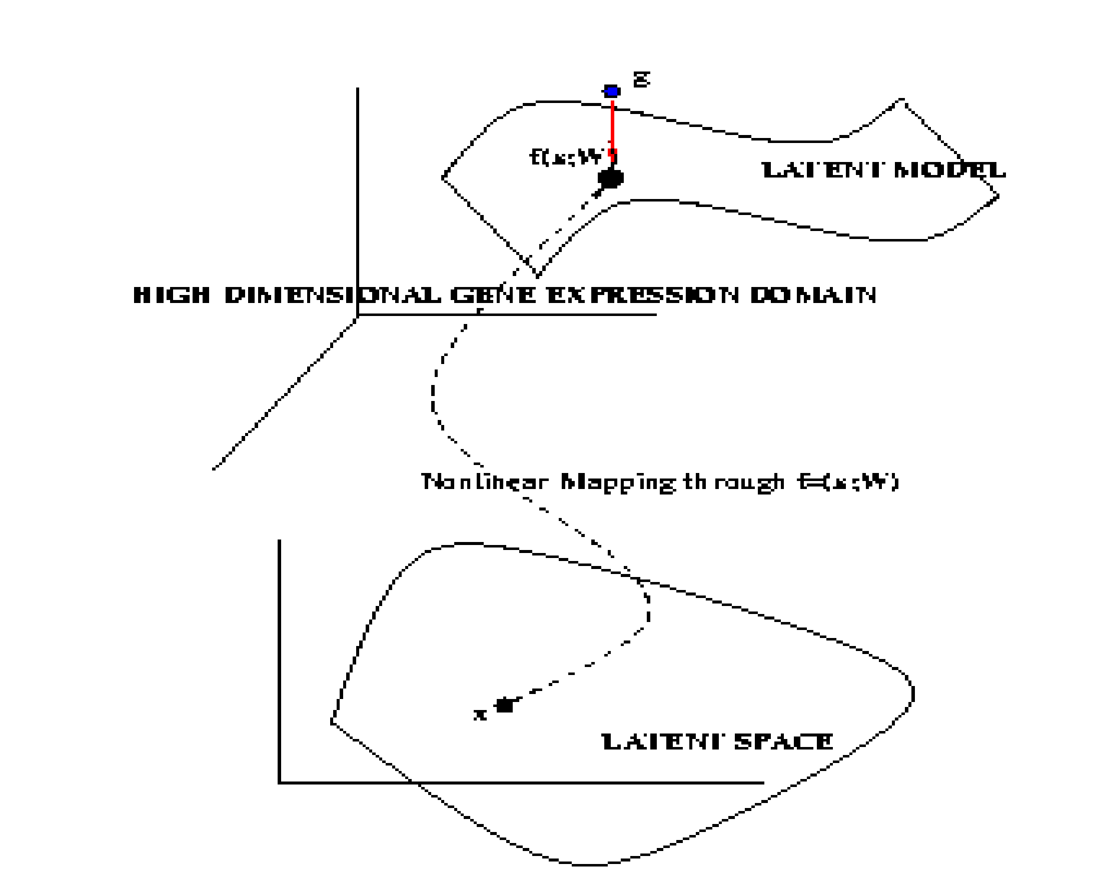
- All network structures were assumed to be equally likely (structure where SEX and SES had parents or/and CP had children are excluded)
- SES has a direct influence on IQ is most suspicious result: new model is considered with a hidden variable pointing SES, IQ or SES, IQ, PE /and none or one or both of (SES-PE, PE-IQ) connections are removed.
- 2x10^{10} times more likely than the best model with no hidden variables.
- Hidden variable is influencing both socioeconomic status and IQ: some measure of ‘parent quality’.

Generative Topographic Mapping (1)

- GTM is a non-linear mapping model between latent space and data space.

$$g = f(x;W) + e$$

$$f(x;W) = \Phi(x)'w$$


g
f(x;W)
LATENT MODEL
HIGH DIMENSIONAL GENE EXPRESSION DOMAIN
Nonlinear Mapping through f(x;W)
x
LATENT SPACE

Generative Topographic Mapping (2)

- A complex data structure is modeled from an intrinsic latent space through a nonlinear mapping.

$$\boxed{t = \Phi(x)W + E}$$

- t : data point
- x : latent point
- Φ : matrix of basis functions
- W : constant matrix
- E : Gaussian noise

Generative Topographic Mapping (3)

- A distribution of x induces a probability distribution in the data space for non-linear y(x,w).

$$\begin{aligned} p(\mathbf{t}|\mathbf{x},\mathbf{W},\beta) &= \mathcal{N}(\mathbf{y}(\mathbf{x},\mathbf{W}),\beta) \\ &= \left(\frac{\beta}{2\pi}\right)^{-D/2} \exp\left\{-\frac{\beta}{2}\sum_{d}^{D}(t_d - y_d(\mathbf{x},\mathbf{W}))^2\right\} \end{aligned}$$

- Likelihood for the grid of K points

$$p(\mathbf{x}) = \frac{1}{K}\sum_{k}^{K}\delta(\mathbf{x}-\mathbf{x}_k),$$

$$p(\mathbf{t}|\mathbf{W},\beta) = \frac{1}{K}\sum_{k}^{K}p(\mathbf{t}|\mathbf{x}_k,\mathbf{W},\beta).$$

Generative Topographic Mapping(4)

- Usually the latent distribution is assumed to be uniform (Grid).
- Each data point is assigned to a grid point probabilistically.
- Data can be visualized by projecting each data point onto the latent space to reveal interesting features
- EM algorithm for training.
  - **Initialize** parameter *W* for a given grid and basis function set.
  - (E-Step) Assign each data point’s probability of belonging to each grid point.
  - (M-Step) Estimate the parameter *W* by maximizing the corresponding log likelihood of data.
  - **Until** some convergence criterion is met.

K-Nearest Neighbor Learning

- Instance
  - points in the *n*-dimensional space $\Re^n$
  - feature vector $<a_1(x), a_2(x),\ldots,a_n(x)>$
- distance

$$d(x_i, x_j) = \sqrt{\sum_{r=1}^{n} (a_r(x_i) - a_r(x_j))^2}$$

- target function : discrete or real value

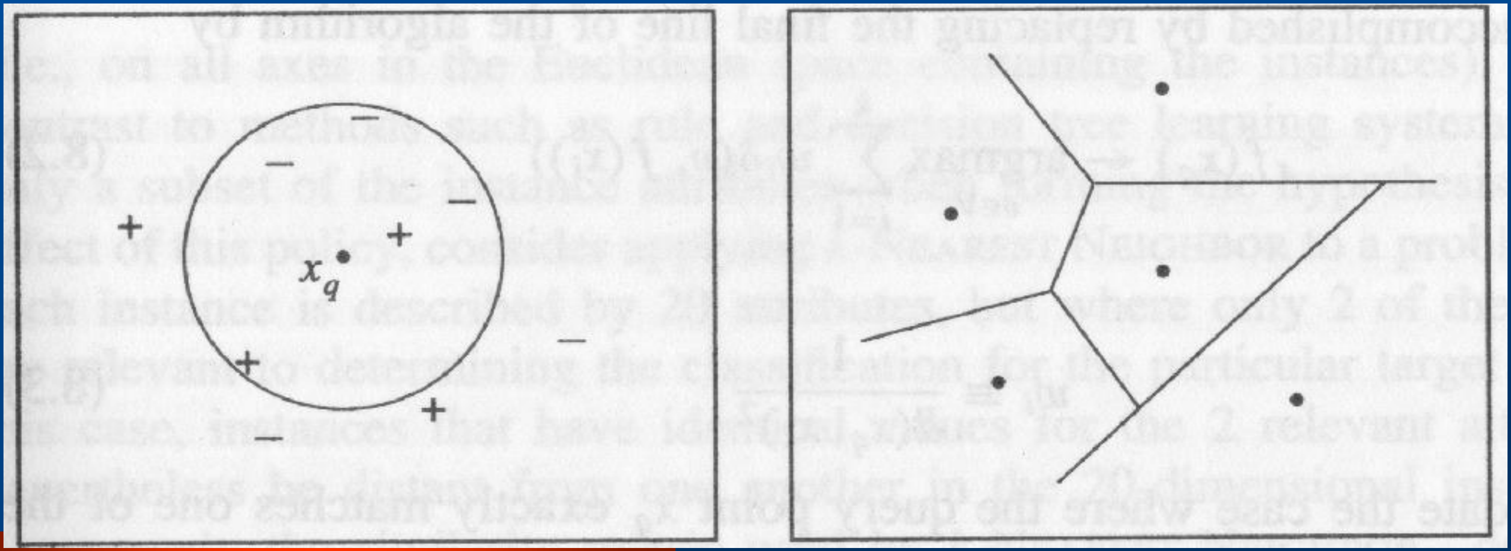
$$f : \Re^n \rightarrow V$$

$$\hat{f}(x_q) \leftarrow \frac{\sum_{i=1}^{k} f(xi)}{k}$$

- Training algorithm:
  - For each training example (x,f(x)), add the example to the list *training_examples*
- Classification algorithm:
  - Given a query instance x_q to be classified,
    - Lex $x_1 \ldots x_k$ denote the k instances from *training_examples* that are nearest to x_q
    - Return

$$\hat{f}(x_q) \leftarrow \arg\max_{v \in V} \sum_{i=1}^{k} \delta(v, f(x_i))$$

$$\text{where } \delta(\mathrm{a}, \mathrm{b}) = 1 \text{ if } \mathrm{a} = \mathrm{b} \text{ and where } \delta(\mathrm{a}, \mathrm{b}) = 0 \text{ otherwise}$$



Distance-Weighted N-N Algorithm

- Giving greater weight to closer neighbors
  - discrete case

$$\hat{f}(x_q) \leftarrow \arg\max_{v \in V} \sum_{i=1}^{k} w_i \delta(v, f(x_i))$$

$$w_i = \frac{1}{d(x_q, x_i)^2}$$

  - real case

$$\hat{f}(x_q) \leftarrow \frac{\sum_{i=1}^{k} w_i f(xi)}{\sum_{i=1}^{k} w_i}$$

Remarks on *k-N-N* Algorithm

- Robust to noisy training data
- Effective in sufficiently large set of training data
- Subset of instance attributes
- Dominated by irrelevant attributes
  - weight each attribute differently
- Indexing the stored training examples
  - *kd*-tree

Radial Basis Functions

- Distance weighted regression and ANN

$$\hat{f}(x) = w_0 + \sum_{u=1}^{k} w_u K_u(d(x_u, x))$$

  - where *xu* : instance from X
  - *Ku*(*d*(*xu*,*x*)) : *kernel function*

- The contribution from each of the *Ku(d(xu,x))* terms is localized to a region nearby the point *xu*   : Gaussian Function

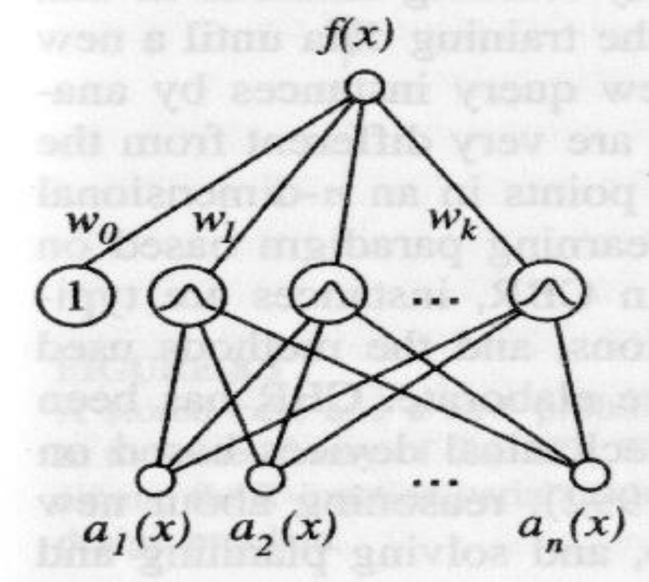
- Corresponding two layer network
  - first layer : computes the values of the various *Ku(d(xu,x))*
  - second layer : computes a linear combination of first-layer unit values.

RBF network

- Training
  - construct kernel function
  - adjust weights

$\hat{f}(x)$: global approximation to $f(x)$

$K_u(d(x_u, x))$ terms is localized to x_u



- RBF networks provide a global approximation to the target function, represented by a linear combination of many local kernel functions.

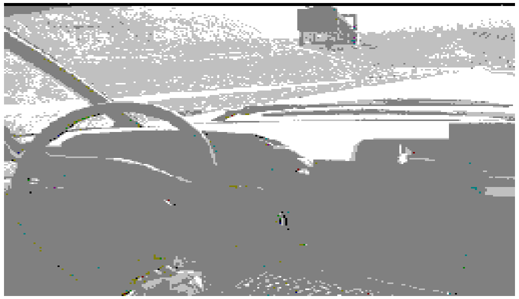
Artificial Neural Networks

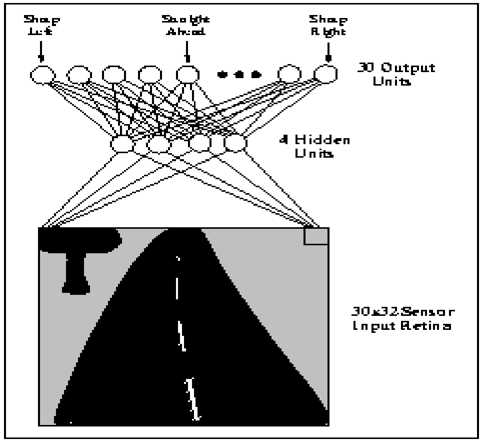
- Artificial neural network(ANN)
  - General, practical method for learning real-valued, discrete-valued, vector-valued functions from examples
- BACPROPAGATION 알고리즘
  - Use gradient descent to tune network parameters to best fit a training set of input-output pairs
- ANN learning
  - Training example의 error에 강하다.
  - Interpreting visual scenes, speech recognition, learning robot control strategy

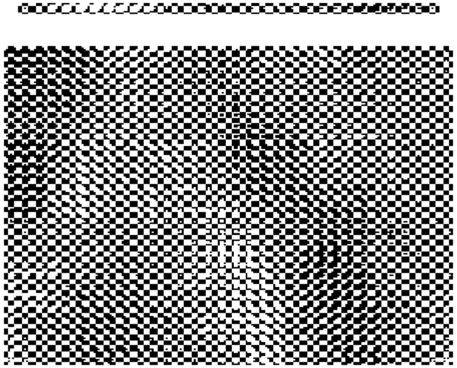
Biological motivation

- 생물학적인 뉴런과의 유사성
  - For 10^{11} neurons interconnected with 10^{4} neurons, 10^{-3} switching times (slower than 10^{-10} of computer), it takes only 10^{-1} to recognize.
  - 병렬 계산(parallel computing)
  - 분산 표현(distributed representation)
- 생물학적인 뉴런과의 차이점
  - 각 뉴런의 출력: single constant vs complex time series of spikes

ALVINN system

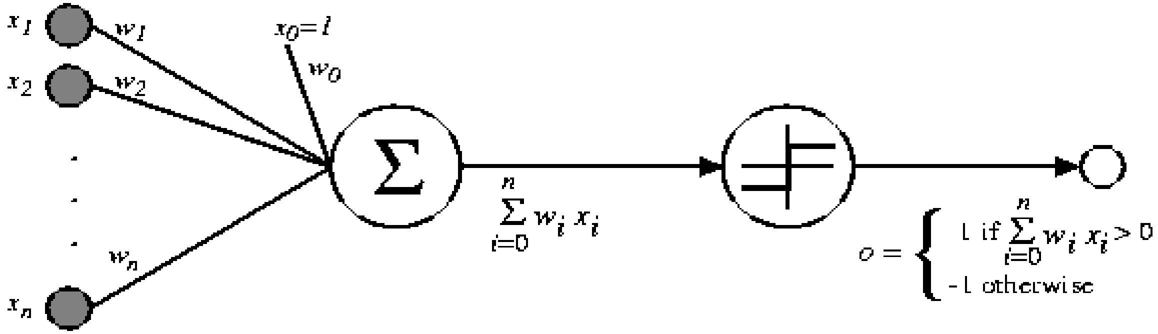






- Input: 30 x 32 grid of pixel intensities (960 nodes)
- 4 hidden units
- Output: direction of steering (30 units)
- Training: 5 min. of human driving
- Test: up to 70 miles for distances of 90 miles on public highway. (driving in the left lane with other vehicles present)

Perceptrons

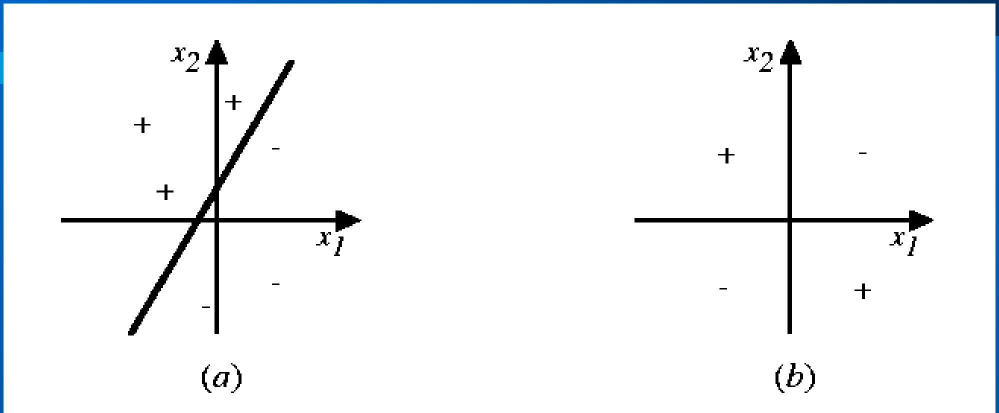


x_1
x_2
x_n
w_1
w_2
w_n
$x_0=1$
w_0
Σ
$\sum_{i=0}^{n} w_i x_i$
$o = \begin{cases} 1 \text{ if } \sum_{i=0}^{n} w_i x_i > 0 \\ -1 \text{ otherwise} \end{cases}$

$$o(x_1, \ldots, x_n) = \begin{cases} 1 & \text{if } w_0 + w_1 x_1 + \cdots + w_n x_n > 0 \\ -1 & \text{otherwise.} \end{cases}$$

- vector of real-valued input
- weights & threshold
- learning: choosing values for the weights

Perceptron의 표현력



(*a*) (*b*)

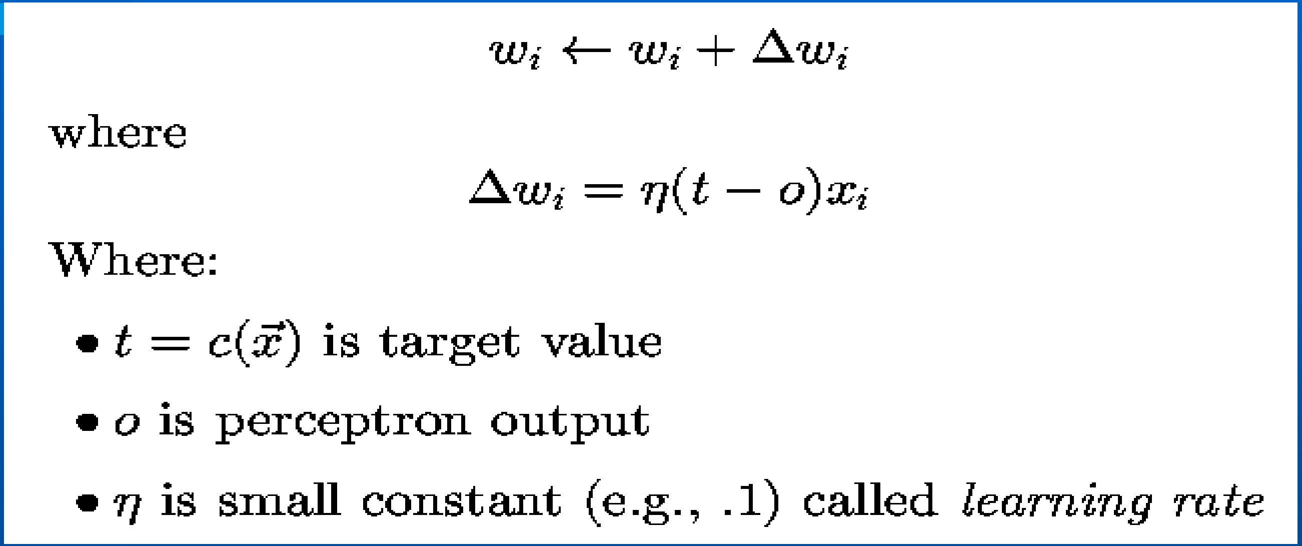
- Hyperplane decision surface for linearly separable example
- many boolean functions(XOR 제외):

(e.g.) AND : w1=w2=1/2, w0=-0.8

OR : w1=w2=1/2, w0=-0.3

- m-of-n function
- disjunctive normal form (disjunction (OR) of a set of conjuctions (AND))

Perceptron rule



$$w_i \leftarrow w_i + \Delta w_i$$

where

$$\Delta w_i = \eta(t - o)x_i$$

Where:

- $t = c(\vec{x})$ is target value
- o is perceptron output
- η is small constant (e.g., .1) called *learning rate*

- 유한번의 학습 후 올바른 가중치를 찾아내려면 충족되어야 할 사항
  - training example이 linearly separable
  - 충분히 작은 learning rate

Gradient descent & Delta rule

- Perceptron rule fails to converge for linearly non-separable examples
- Delta rule can overcome the difficulty of perceptron rule by using gradient descent
- In the training of unthresholded perceptron.

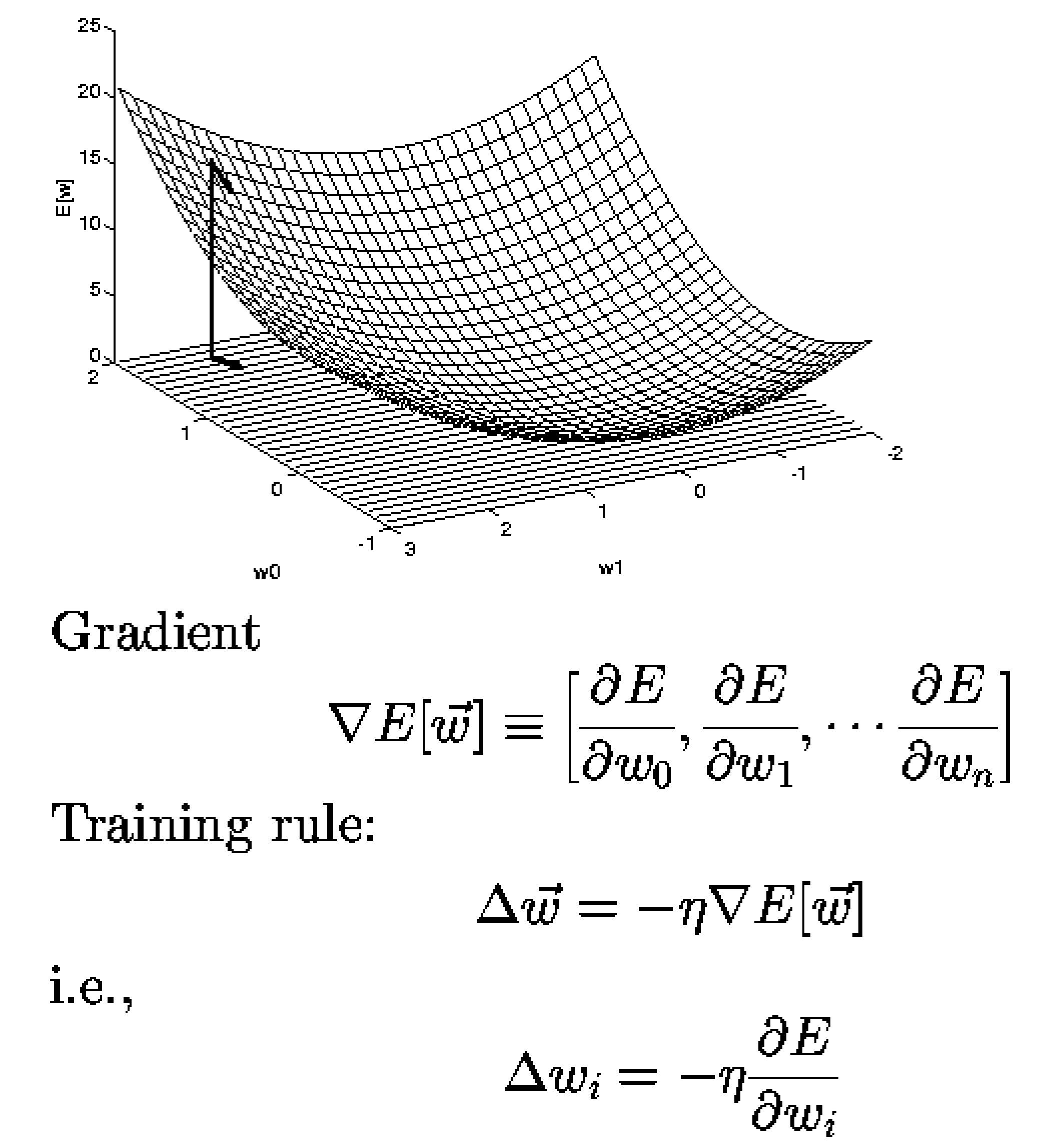
$$o(\vec{x}) = \vec{w} \cdot \vec{x}$$

training error is given as a function of weights:

$$E(\vec{w}) = \frac{1}{2} \sum_{d \in D} (t_d - o_d)^2$$

- Gradient descent can search the hypothesis space of different types of continuously parameterized hypotheses.

Hypethesis space



Gradient

$$\nabla E[\vec{w}] \equiv \left[\frac{\partial E}{\partial w_0}, \frac{\partial E}{\partial w_1}, \cdots \frac{\partial E}{\partial w_n}\right]$$

Training rule:

$$\Delta \vec{w} = -\eta \nabla E[\vec{w}]$$

i.e.,

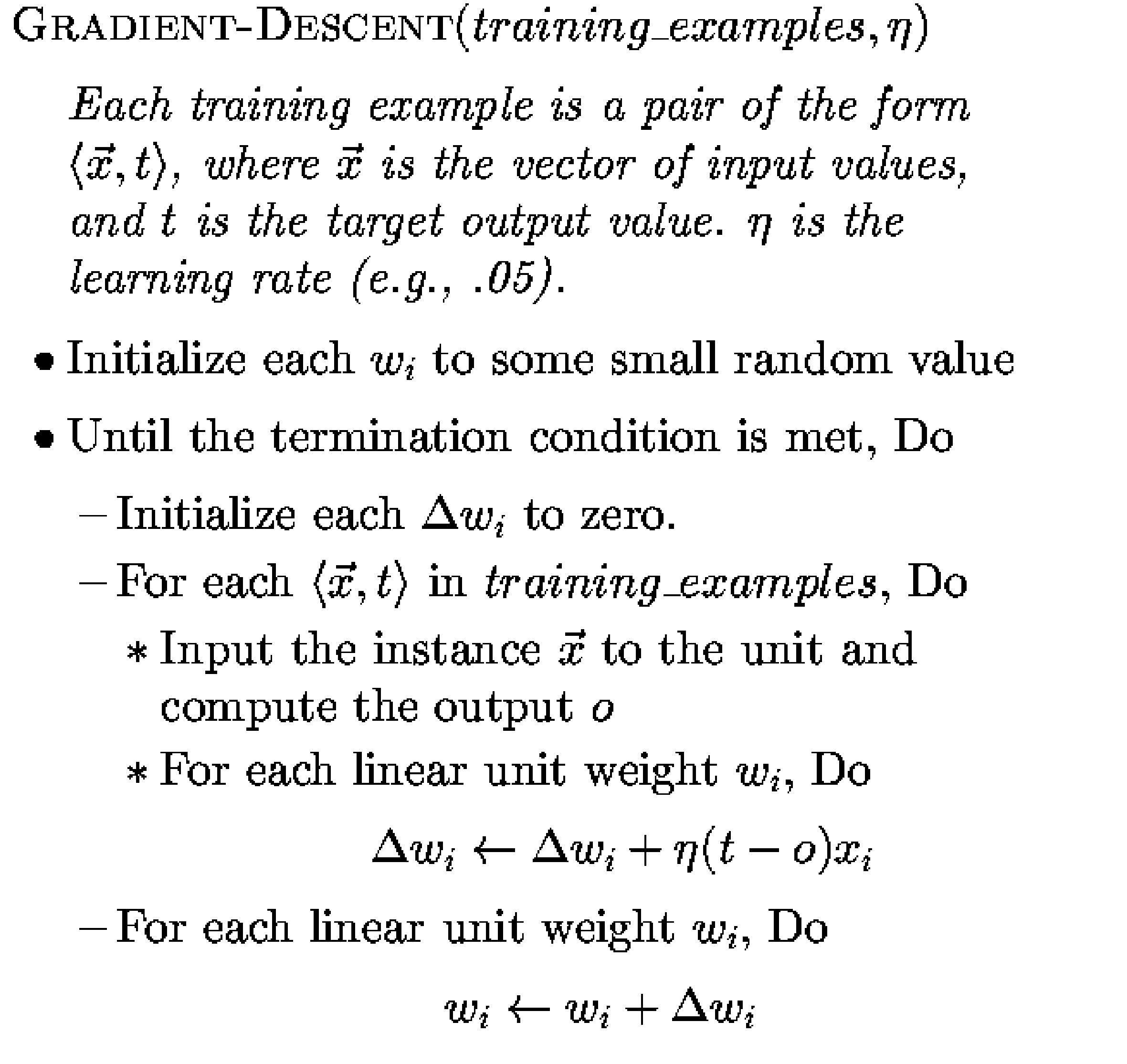
$$\Delta w_i = -\eta \frac{\partial E}{\partial w_i}$$

Gradient descent

$$
\begin{aligned}
\frac{\partial E}{\partial w_i} &= \frac{\partial}{\partial w_i}\frac{1}{2}\sum_d (t_d - o_d)^2 \\
&= \frac{1}{2}\sum_d \frac{\partial}{\partial w_i}(t_d - o_d)^2 \\
&= \frac{1}{2}\sum_d 2(t_d - o_d)\frac{\partial}{\partial w_i}(t_d - o_d) \\
&= \sum_d (t_d - o_d)\frac{\partial}{\partial w_i}(t_d - \vec{w}\cdot\vec{x_d}) \\
\frac{\partial E}{\partial w_i} &= \sum_d (t_d - o_d)(-x_{i,d})
\end{aligned}
$$

- gradient: steepest increase in E

$$\Delta w_i = \eta \sum_{d \in D} (t_d - o_d) x_{id}$$



```
GRADIENT-DESCENT($training\_examples, \eta$)

    Each training example is a pair of the form
    $\langle \vec{x}, t \rangle$, where $\vec{x}$ is the vector of input values,
    and $t$ is the target output value. $\eta$ is the
    learning rate (e.g., .05).

• Initialize each $w_i$ to some small random value
• Until the termination condition is met, Do
    – Initialize each $\Delta w_i$ to zero.
    – For each $\langle \vec{x}, t \rangle$ in $training\_examples$, Do
        ∗ Input the instance $\vec{x}$ to the unit and
          compute the output $o$
        ∗ For each linear unit weight $w_i$, Do
              $\Delta w_i \leftarrow \Delta w_i + \eta(t - o)x_i$
    – For each linear unit weight $w_i$, Do
              $w_i \leftarrow w_i + \Delta w_i$
```

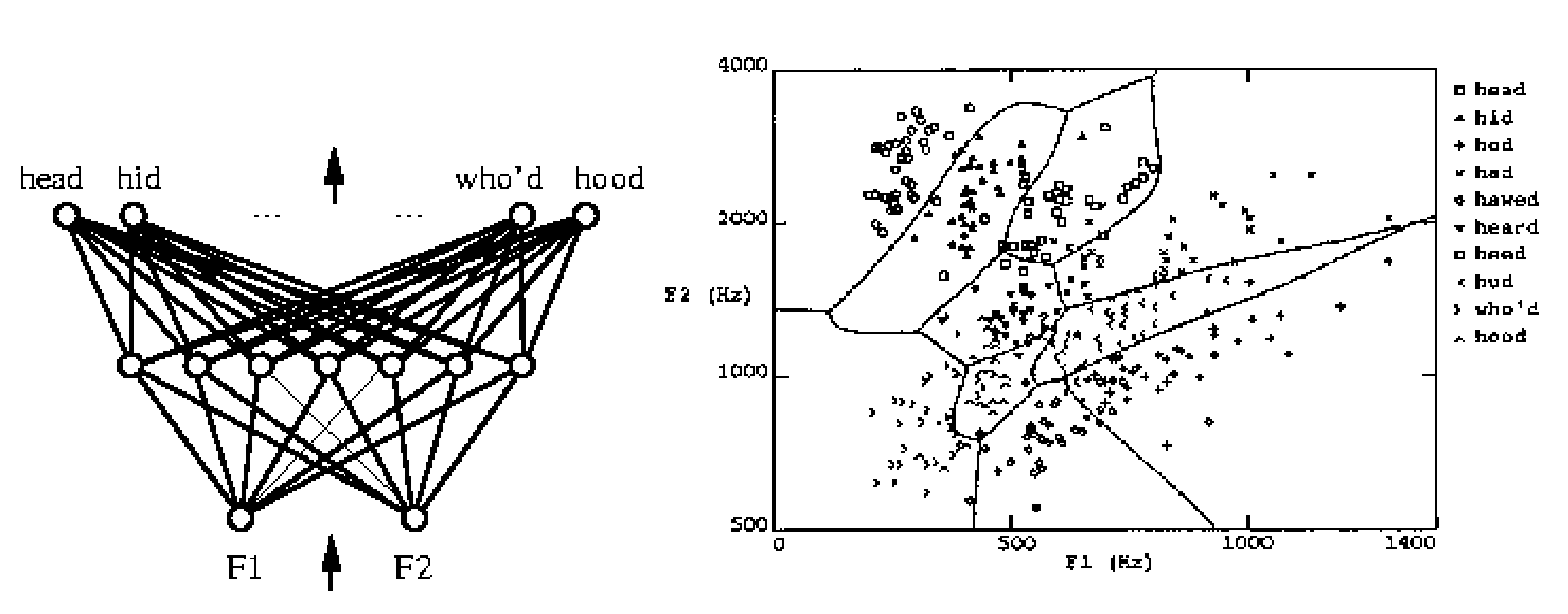
Gradient descent(cont'd)

- Training example의 linearly separable 여부에 관계없이 하나의 global minimum을 찾는다.
- Learning rate가 큰 경우 overstepping의 문제 -> learning rate를 점진적으로 줄이는 방법을 사용하기도 한다.

Remark

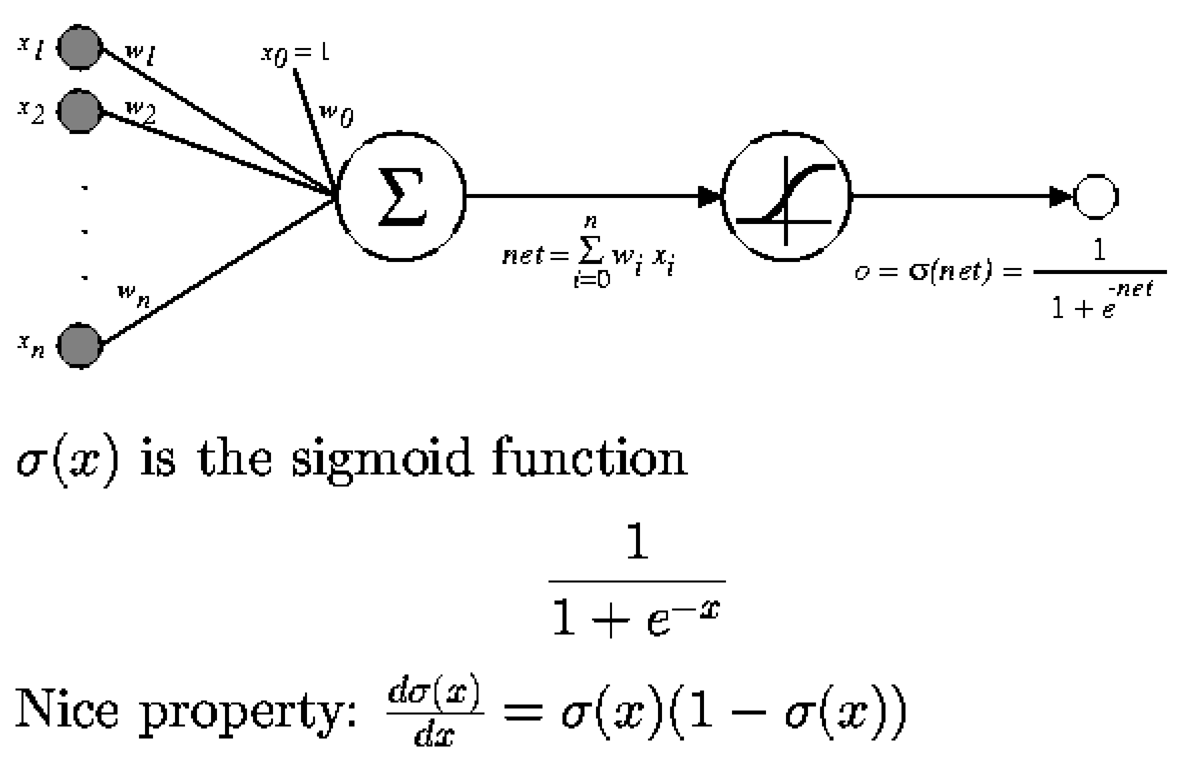
- Perceptron rule
  - thresholded output
  - 정확한 weight (perfect classification)
  - linearly separable
- Delta rule
  - unthresholded output
  - 점근적으로 에러를 최소화하는 weight
  - non-linearly separable

Multilayer networks



- Nonlinear decision surface
- Multiple layers of linear units still produce only linear functions
- Perceptron's output is not differentiable wrt. inputs

Differential threshold unit


x_1
x_2
x_n
w_1
w_2
w_n
$x_0 = 1$
w_0
Σ
$net = \sum_{i=0}^{n} w_i x_i$
$o = \sigma(net) = \frac{1}{1+e^{-net}}$

$\sigma(x)$ is the sigmoid function

$$\frac{1}{1+e^{-x}}$$

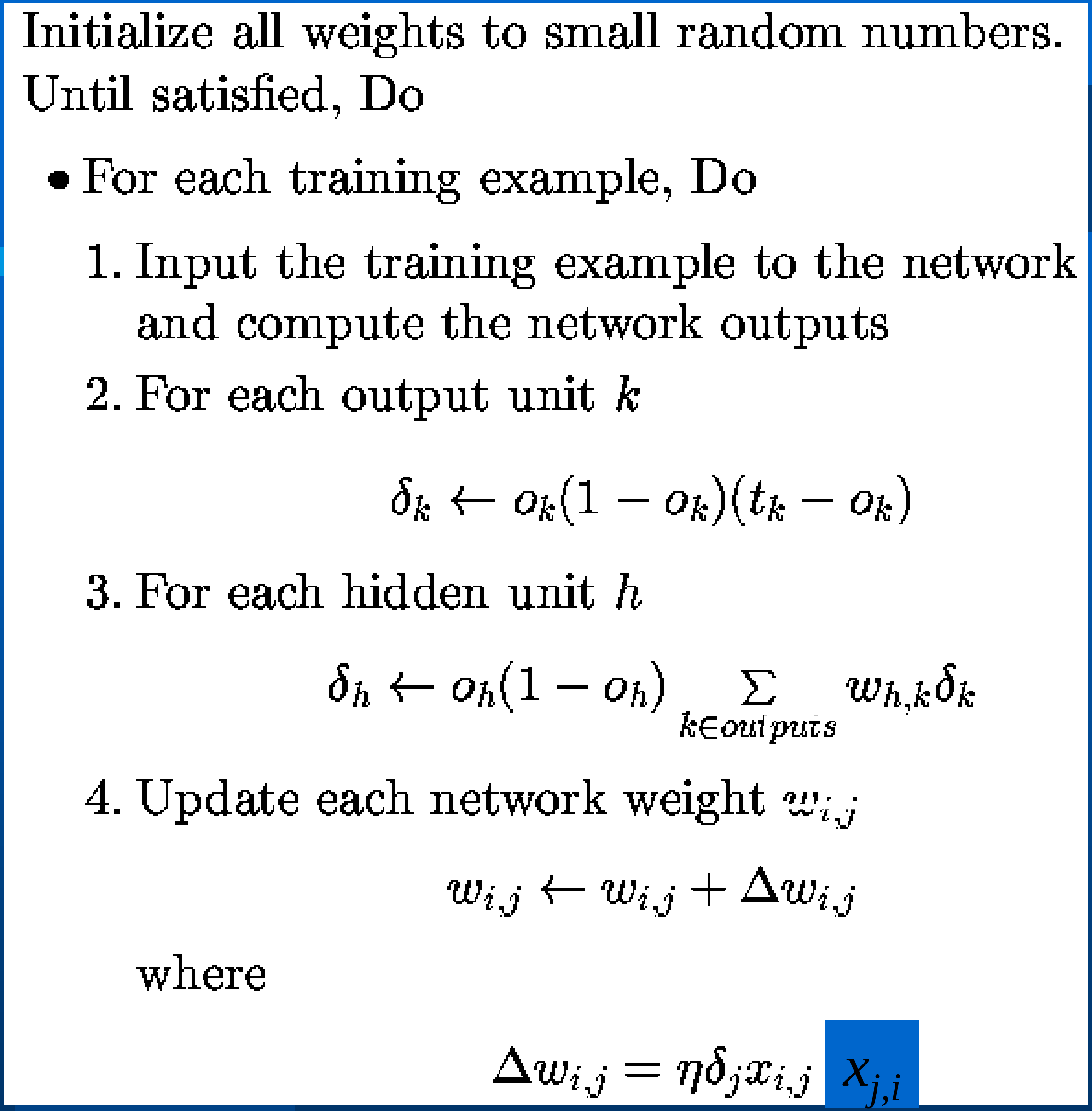
Nice property: $\frac{d\sigma(x)}{dx} = \sigma(x)(1-\sigma(x))$

- Sigmoid function
  - nonlinear, differentiable

BACKPROPAGATION 알고리즘

- Backpropagation algorithm learns the weights of multi-layer network by minimizing the squared error between network output values and target values employing gradient descent.
- For multiple outputs, the errors are sum of all the output errors.

$$E(\vec{w}) \equiv \frac{1}{2} \sum_{d \in D} \sum_{k \in outputs} (t_{kd} - o_{kd})^2$$



```
Initialize all weights to small random numbers.
Until satisfied, Do
  • For each training example, Do
    1. Input the training example to the network
       and compute the network outputs
    2. For each output unit $k$
           $\delta_k \leftarrow o_k(1 - o_k)(t_k - o_k)$
    3. For each hidden unit $h$
           $\delta_h \leftarrow o_h(1 - o_h) \sum_{k \in outputs} w_{h,k}\delta_k$
    4. Update each network weight $w_{i,j}$
           $w_{i,j} \leftarrow w_{i,j} + \Delta w_{i,j}$
       where
           $\Delta w_{i,j} = \eta \delta_j x_{i,j}$   $x_{j,i}$
```

($x_{j,\,i}$: input from node *i* to node *j*.

δ_j: error-like term on the node *j*)

BACKPROPAGATION 알고리즘(cont’d)

- Multiple local minima
- Termination conditions
  - fixed number of iteration
  - error threshold
  - error of separate validation set

Variations of BACKPROPAGATION 알고리즘

- Adding momentum
  - 직전의 loop에서의 weight 갱신이 영향을 미침

$$\Delta w_{ji}(n) = \eta \delta_j x_{ji} + \alpha \Delta w_{ji}(n-1)$$

- Learning in arbitrary acyclic network

$$\delta_r \leftarrow o_r(1-o_r) \sum_{s \in layer\ m+1} w_{sr}\delta_s \qquad (multilayer)$$

$$\delta_r \leftarrow o_r(1-o_r) \sum_{s \in Downstream(r)} w_{sr}\delta_s \qquad (acyclic)$$

BACKPROPAGATION rule

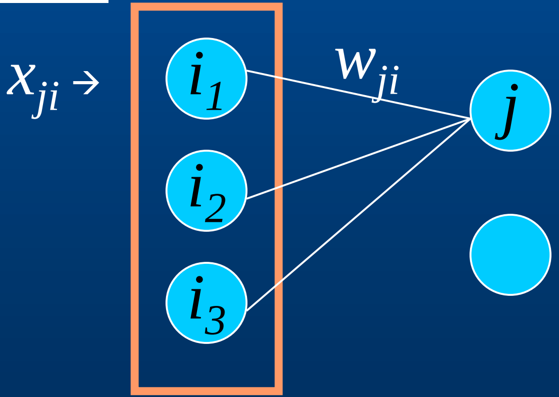
$$\Delta w_{ji} = -\eta \frac{\partial E_d}{\partial w_{ji}} \quad (\textit{update of the weight from input } i \textit{ to unit } j)$$

$$E_d(\vec{w}) \equiv \frac{1}{2} \sum_{k \in outputs} (t_k - o_k)^2 \quad (\textit{the error on training example } d)$$

$$(net)_j = \sum_i w_{ji} x_{ji} \quad (\textit{the weighted sum of inputs for unit } j)$$

$$(e.g. \qquad o_j = \sum_i w_{ji} x_{ji})$$

$$\frac{\partial E_d}{\partial w_{ji}} = \frac{\partial E_d}{\partial net_j} \frac{\partial net_j}{\partial w_{ji}} = \frac{\partial E_d}{\partial net_j} x_{ji}$$



- **Training rule for output unit**

$$\frac{\partial E_d}{\partial net_j} = \frac{\partial E_d}{\partial o_j} \frac{\partial o_j}{\partial net_j}$$

$$\frac{\partial E_d}{\partial o_j} = \frac{\partial}{\partial o_j} \frac{1}{2} \sum_{k \in outputs} (t_k - o_k)^2$$

$$\frac{\partial E_d}{\partial o_j} = \frac{\partial}{\partial o_j} \frac{1}{2} (t_j - o_j)^2 = \frac{1}{2} 2(t_j - o_j) \frac{\partial (t_j - o_j)}{\partial o_j} = -(t_j - o_j)$$

$$\frac{\partial o_j}{\partial net_j} = \frac{\partial \sigma(net_j)}{\partial net_j} = o_j(1 - o_j) \qquad (\sigma(y) = 1/(1 + e^{-y}))$$

$$\frac{\partial E_d}{\partial net_j} = \delta_j$$

$$\Delta w_{ji} = -\eta \frac{\partial E_d}{\partial w_{ji}} = \eta(t_j - o_j)o_j(1 - o_j)x_{ji} = \eta \delta_j x_{ji}$$

- **Training rule for hidden unit**

$$\frac{\partial E_d}{\partial net_j} = \sum_{k\in Downstream(j)} \frac{\partial E_d}{\partial net_k}\frac{\partial net_k}{\partial net_j} = \sum_{k\in Downstream(j)} -\delta_k \frac{\partial net_k}{\partial net_j}$$

$$= \sum_{k\in Downstream(j)} -\delta_k \frac{\partial net_k}{\partial o_j}\frac{\partial o_j}{\partial net_j} = \sum_{k\in Downstream(j)} -\delta_k w_{kj} \frac{\partial o_j}{\partial net_j}$$

$$= \sum_{k\in Downstream(j)} -\delta_k w_{kj} o_j (1-o_j)$$

$$\delta_j = o_j(1-o_j) \sum_{k\in Downstream(j)} \delta_k w_{kj}$$

$$
\begin{aligned}
\frac{\partial E}{\partial w_i} &= \frac{\partial}{\partial w_i}\frac{1}{2}\sum_{d\in D}(t_d-o_d)^2 \\
&= \frac{1}{2}\sum_d \frac{\partial}{\partial w_i}(t_d-o_d)^2 \\
&= \frac{1}{2}\sum_d 2(t_d-o_d)\,\frac{\partial}{\partial w_i}(t_d-o_d) \\
&= \sum_d (t_d-o_d)\left(-\frac{\partial o_d}{\partial w_i}\right) \\
&= -\sum_d (t_d-o_d)\,\frac{\partial o_d}{\partial net_d}\,\frac{\partial net_d}{\partial w_i}
\end{aligned}
$$

But we know:

$$\frac{\partial o_d}{\partial net_d} = \frac{\partial \sigma(net_d)}{\partial net_d} = o_d(1-o_d)$$

$$\frac{\partial net_d}{\partial w_i} = \frac{\partial(\vec{w}\cdot\vec{x}_d)}{\partial w_i} = x_{i,d}$$

So:

$$\frac{\partial E}{\partial w_i} = -\sum_{d\in D}(t_d-o_d)o_d(1-o_d)x_{i,d}$$

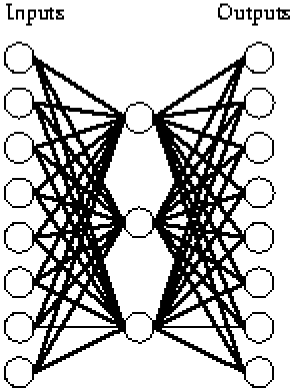
Convergence and local minima

- Only guarantees local minima
  - This problem is not severe
- Algorithm is highly effective
- the more weights, the less severe local minima problem
- If weights are initialized to values near zero, the network will represent very smooth function (almost linear) in its inputs: sigmoid function is approx. linear when the weights are small.
- Common remedies for local minima:
  - Add momentum term to escape the local minima.
  - Use stochastic (incremental) gradient descent: different error surface for each example to prevent getting stuck
  - Training of multiple networks and select the best one over a separate validation data set

Hidden layer representation

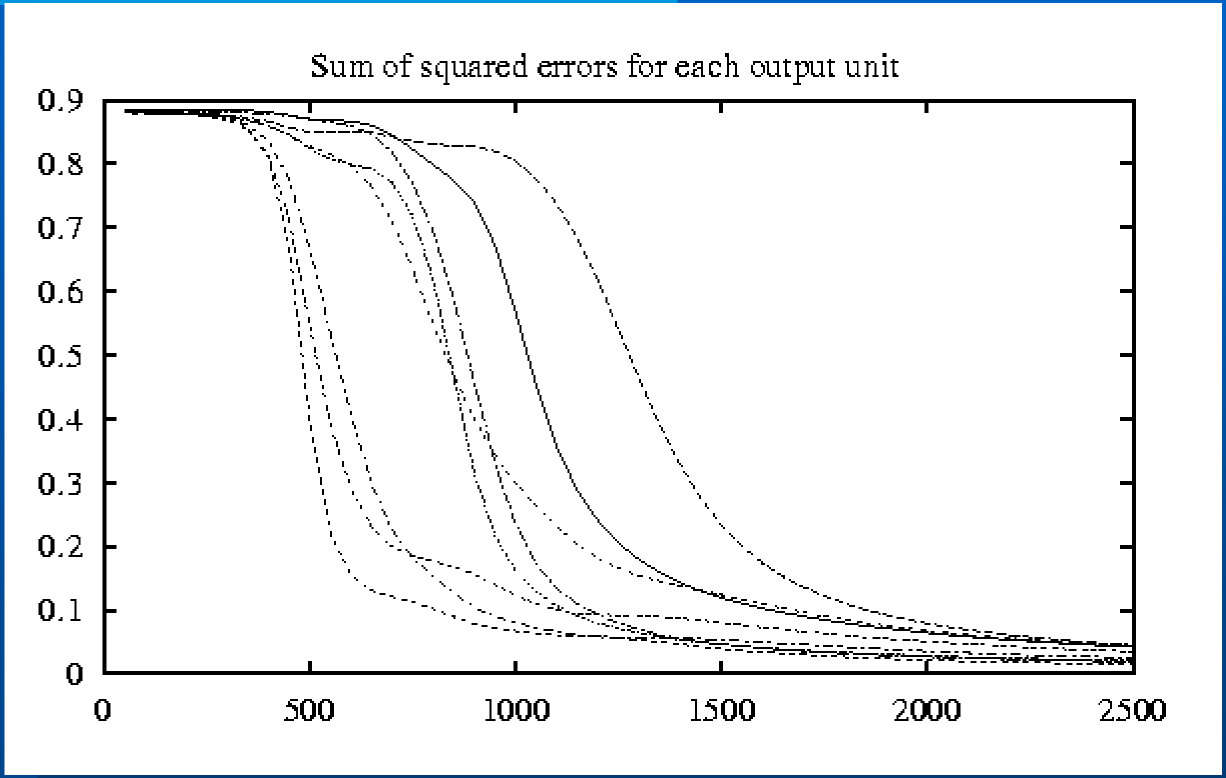
- Automatically discover useful representations at the hidden layers
- Allows the learner to invent features not explicitly introduced by the human designer.

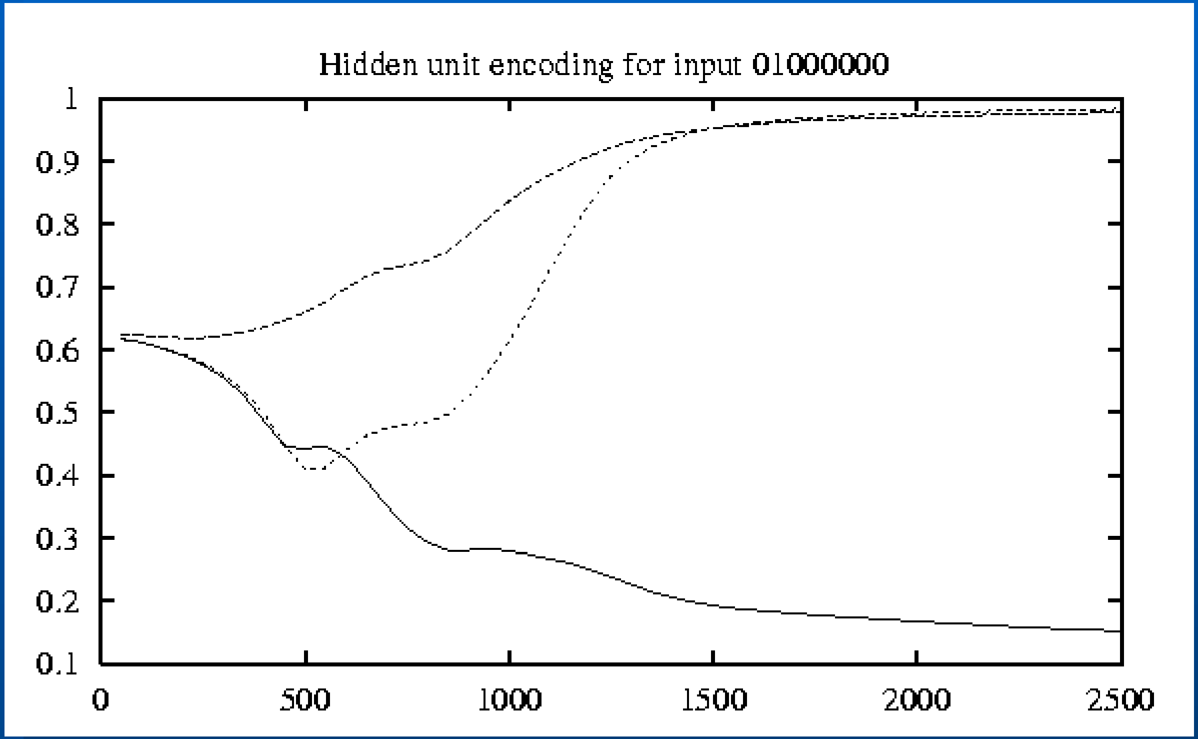
A network:

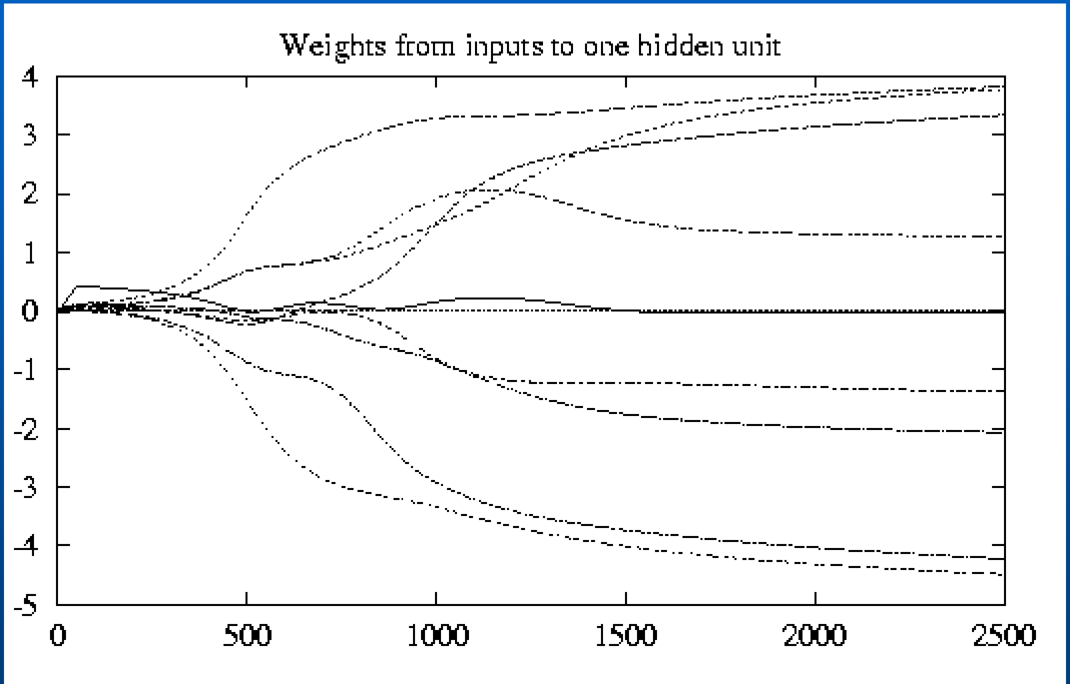


Learned hidden layer representation:

Input		Hidden Values				Output
10000000	→	.89	.04	.08	→	10000000
01000000	→	.01	.11	.88	→	01000000
00100000	→	.01	.97	.27	→	00100000
00010000	→	.99	.97	.71	→	00010000
00001000	→	.03	.05	.02	→	00001000
00000100	→	.22	.99	.99	→	00000100
00000010	→	.80	.01	.98	→	00000010
00000001	→	.60	.94	.01	→	00000001

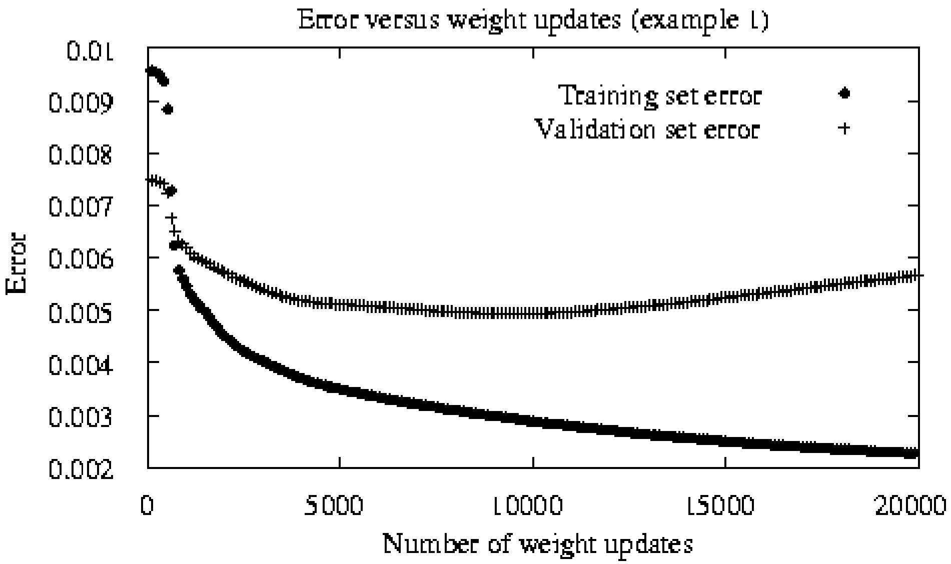

Sum of squared errors for each output unit
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
500
1000
1500
2000
2500

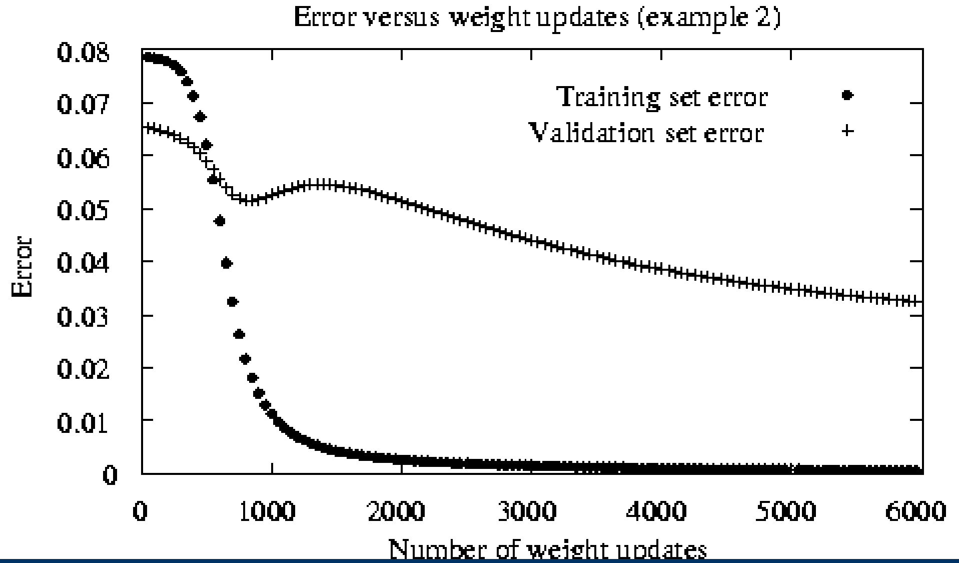

Hidden unit encoding for input 01000000
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
500
1000
1500
2000
2500


Weights from inputs to one hidden unit
4
3
2
1
0
-1
-2
-3
-4
-5
0
500
1000
1500
2000
2500

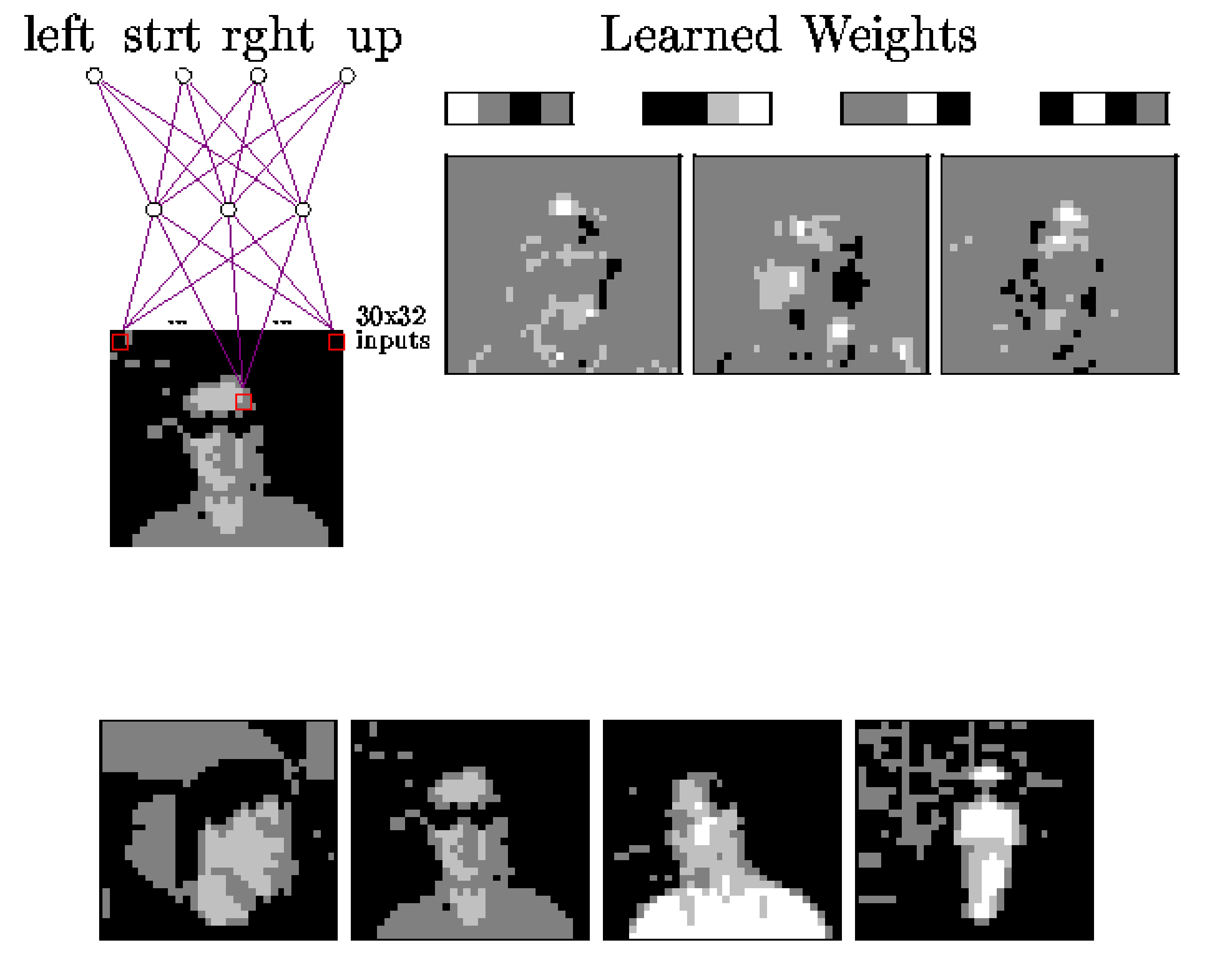
Generalization, overfitting, stopping criterion

- Terminating condition
  - Threshold on the training error: poor strategy
  - Susceptible to overfitting: create overly complex decision surfaces that fit noise in the training data
- Techniques to address the overfitting problem:
- Weight decay: decrease each weight by small factor (equivalent to modifying the definition of error to include a penalty term)
- Cross-validation approach: validation data in addition to the training data (lowest error over the validation set)
- K-fold cross-validation: For small training sets, cross validation is performed k different times and averaged (e.g. training set is partitioned into k subsets and then the mean iteration number is used.)


Error versus weight updates (example 1)


Error versus weight updates (example 2)

Face recognition



- Images of 20 different people/ 32images per person: varying expressions, looking directions, is/is not wearing sunglasses. Also variation in the background, clothing, position of face
- Total of 624 greyscale images. Each input image:120*128 → 30*32 with each pixel intensity from 0 (Black) to 255 (White)
  - Reducing computational demands
  - mean value (cf, ALVINN: random)
- 1-of-n output encoding
  - More degree than single output unit
  - The difference between the highest and second highest valued output can be used as a measure of confidence in the network prediction.
  - Sigmoid units cannot produce extreme values: avoid 0, 1 in the target values. <0.9, 0.1, 0.1, 0.1>
- 2 layers, 3 units -> 90% success

Alternative error functions

- Adding a penalty term for weight magnitude

$$E(\vec{w}) \equiv \frac{1}{2} \sum_{d \in D} \sum_{k \in outputs} (t_{kd} - o_{kd})^2 + \gamma \sum_{i,j} w_{ji}^2$$

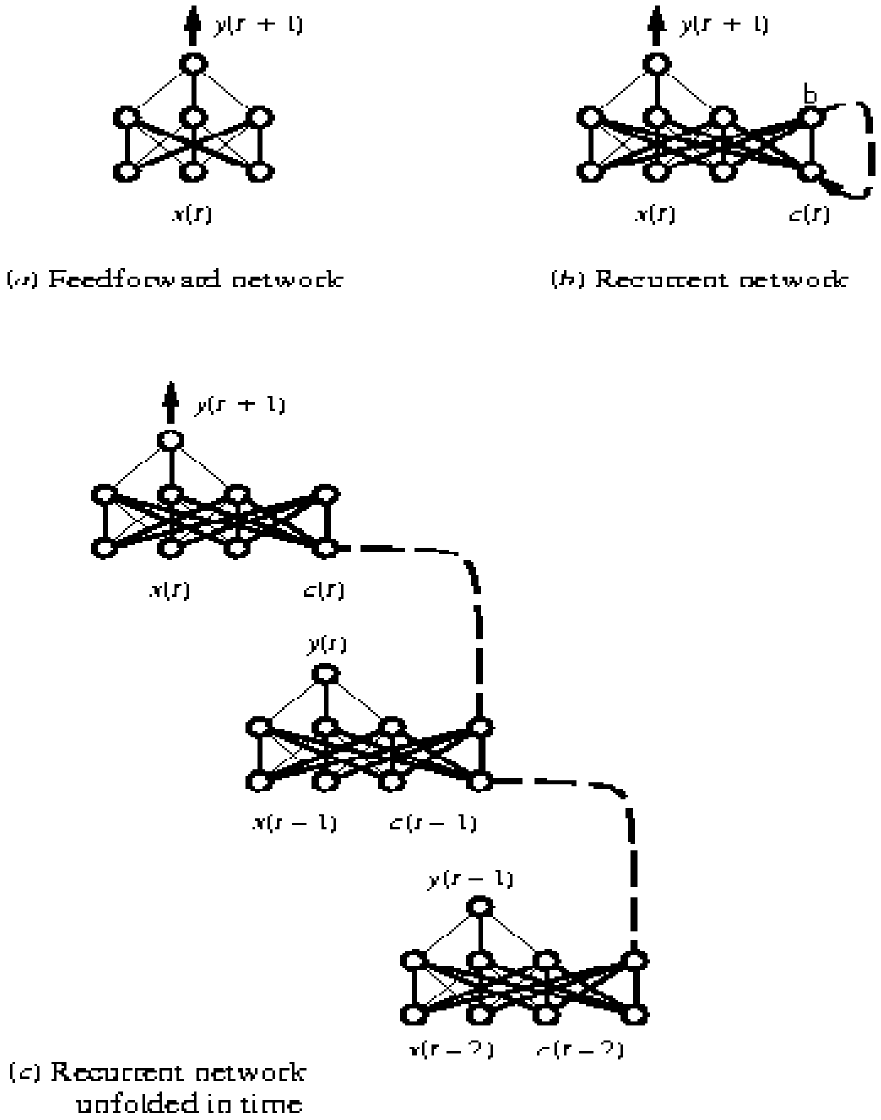
- Adding a derivative of the target function

$$E(\vec{w}) \equiv \frac{1}{2} \sum_{d \in D} \sum_{k \in outputs} \left[(t_{kd} - o_{kd})^2 + \mu \sum_{j \in inputs} \left(\frac{\partial t_{kd}}{\partial x_d^j} - \frac{\partial o_{kd}}{\partial x_d^j} \right)^2 \right.$$

- Minimizing the cross entropy of the network wrt. the target values. (KL divergence: D(t,o)=Σtlog(t/o))

$$-\sum_{d \in D} t_d \log o_d + (1 - t_d) \log(1 - o_d)$$

Recurrent networks



(a) Feedforward network

(b) Recurrent network

(c) Recurrent network
unfolded in time

3. DNA Microarrays

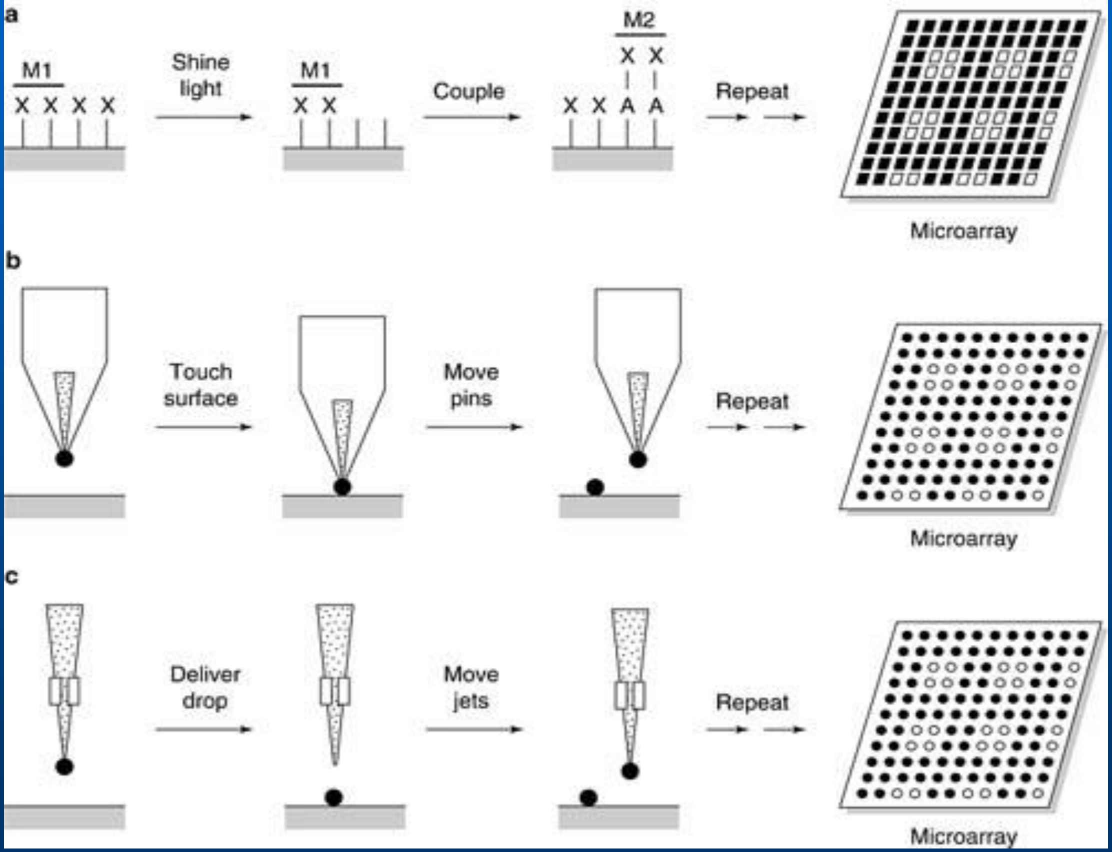
- DNA Chip
  - In the traditional "one gene in one experiment" method, the throughput is very limited and the "whole picture" of gene function is hard to obtain.
  - DNA chip hybridizes thousands of DNA samples of each gene on a glass with special cDNA samples.
  - It promises to monitor the whole genome on a single chip so that researchers can have a better picture of the the interactions among thousands of genes simultaneously.
- Applications of DNA Microarray Technology
  - Gene discovery
  - Disease diagnosis
  - Drug discovery: *Pharmacogenomics*
  - Toxicological research: *Toxicogenomics*

Genes and Life

- It is believed that thousands of genes and their products (i.e., RNA and proteins) in a given living organism function in a complicated and orchestrated way that creates the mystery of life.
- Traditional methods in molecular biology work on a “one gene in one experiment” basis.
- Recent advance in DNA microarrays or DNA chips technology makes it possible to measure the expression levels of thousands of genes simultaneously.

DNA Microarray Technology

- Photolithoraphy methods (a)
- Pin microarray methods (b)
- Inkjet methods (c)
- Electronic array methods



Analysis of DNA Microarray Data

Previous Work

- Characteristics of data
  - Analysis of expression ratio based on each sample
  - Analysis of time-variant data
- Clustering
  - Self-organizing maps [Golub et al., 1999]
  - Singular value decomposition [Orly Alter et al., 2000]
- Classification
  - Support vector machines [Brown et al., 2000]
- Gene identification
  - Information theory [Stefanie et al., 2000]
- Gene modeling
  - Bayesian networks [Friedman et.al., 2000]

DNA Microarray Data Mining

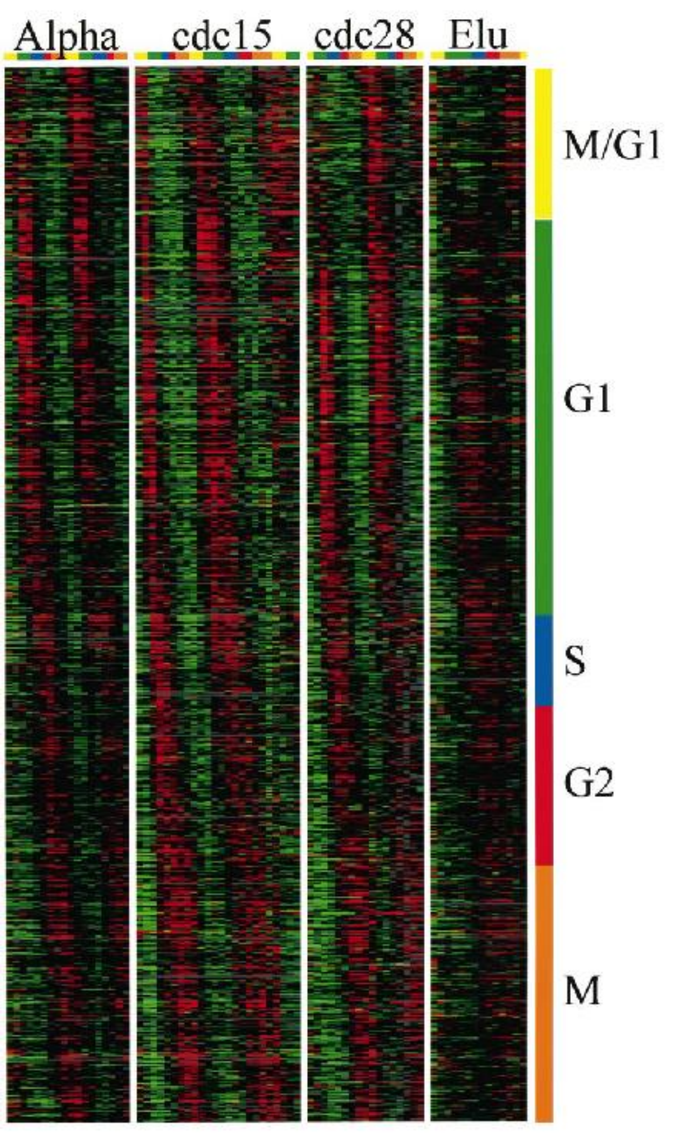
- Clustering
  - Find some groups of genes that show the similar pattern in some conditions.
  - PCA
  - SOM
- Genetic network analysis
  - Determine the regulatory interactions between genes and their derivatives.
  - Linear models
  - Neural networks
  - Probabilistic graphical models

CAMDA-2000 Data Sets

- CAMDA
  - Critical Assessment of Techniques for Microarray Data Mining
  - Purpose: Evaluate the data-mining techniques available to the microarray community.
- Data Set 1
  - Identification of cell cycle-regulated genes
  - Yeast Sacchromyces cerevisiae by microarray hybridization.
  - Gene expression data with 6,278 genes.
- Data Set 2
  - Cancer class discovery and prediction by gene expression monitoring.
  - Two types of cancers: acute myeloid leukemia (AML) and acute lymphoblastic leukemia (ALL).
  - Gene expression data with 7,129 genes.

CAMDA-2000 Data Set 1

Identification of Cell Cycle-regulated Genes of the Yeast by Microarray Hybridization

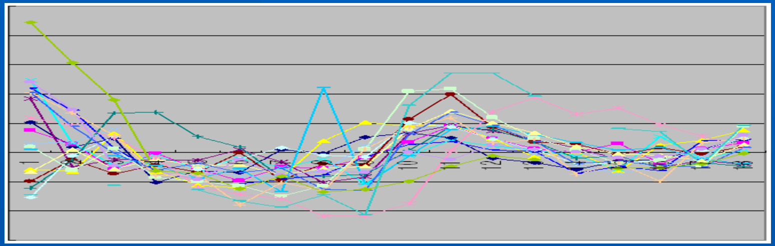


- Data given: gene expression levels of 6,278 genes spanned by time
  - α Factor-based synchronization: every 7 minute from 0 to 119 (18)
  - Cdc15-based synchronization: every 10 minute from 10 to 290 (24)
  - Cdc28-based synchronization: every 10 minute from 0 to 160 (17)
  - Elutriation (size-based synchronization): every 30 minutes from 0 to 390 (14)
- Among 6,278 genes
  - 104 genes are known to be cell-cycle regulated
    - classified into: M/G1 boundary (19), late G1 SCB regulated (14), late G1 MCB regulated (39), S-phase (8), S/G2 phase (9), G2/M phase (15).
  - 250 cell cycle–regulated genes might exist

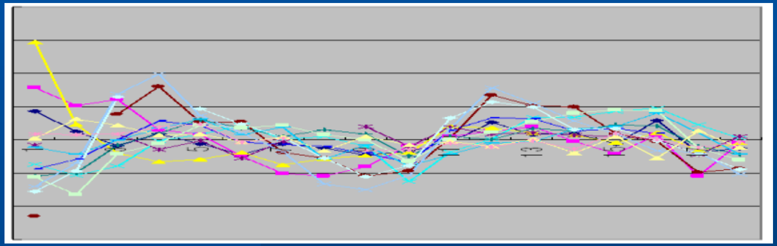
CAMDA-2000 Data Set 1

Characteristics of data (α Factor-based Synchronization)

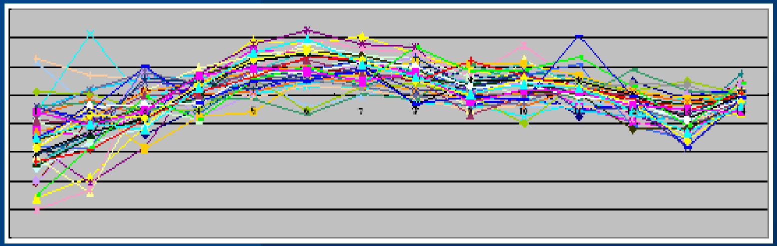
- M/G1 boundary



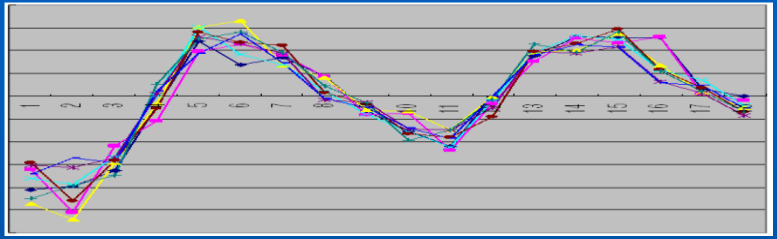
- Late G1 SCB regulated



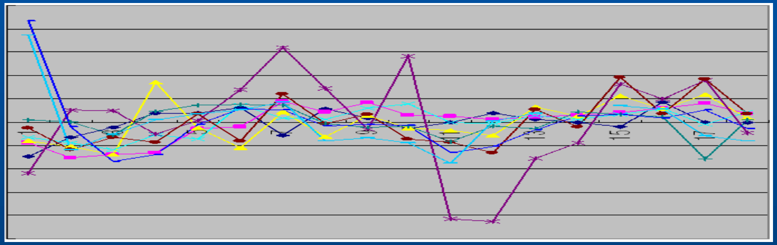
- Late G1 MCB regulated



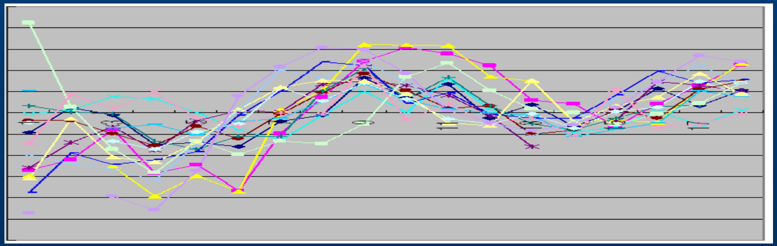
- S Phase



- S/G2 Phase

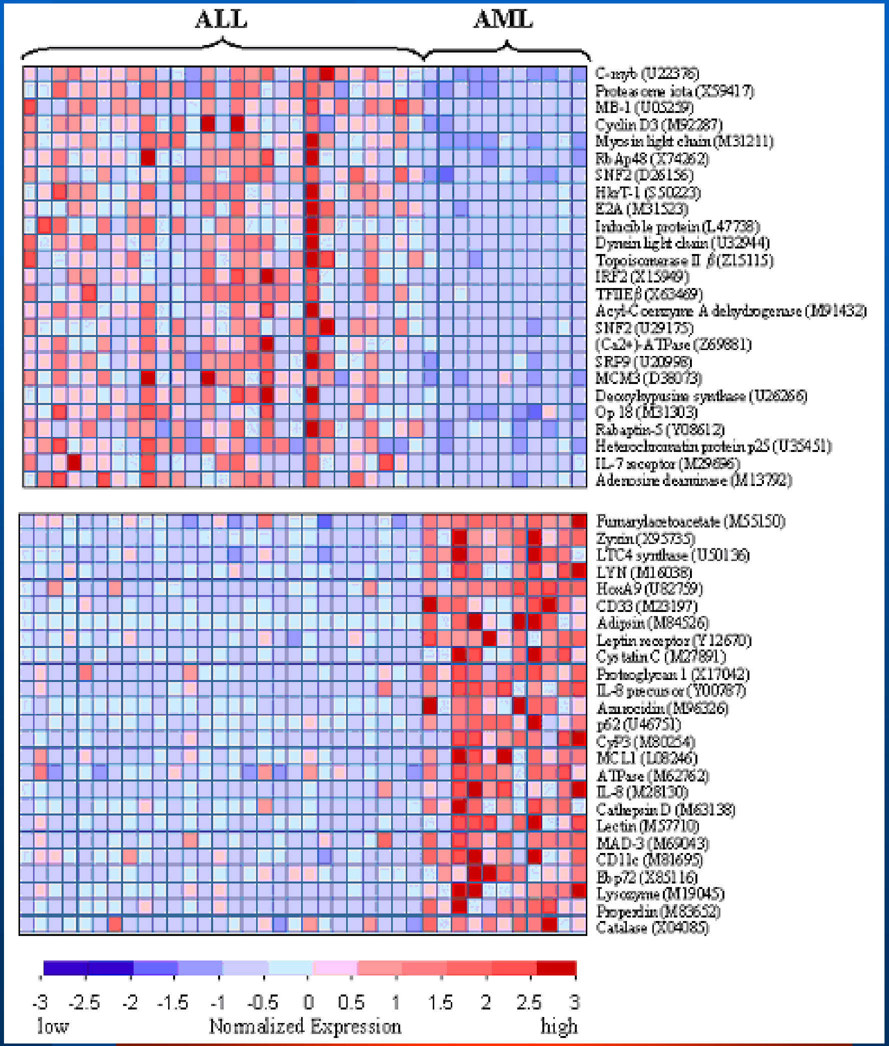


- G2/M Phase



CAMDA-2000 Data Set 2

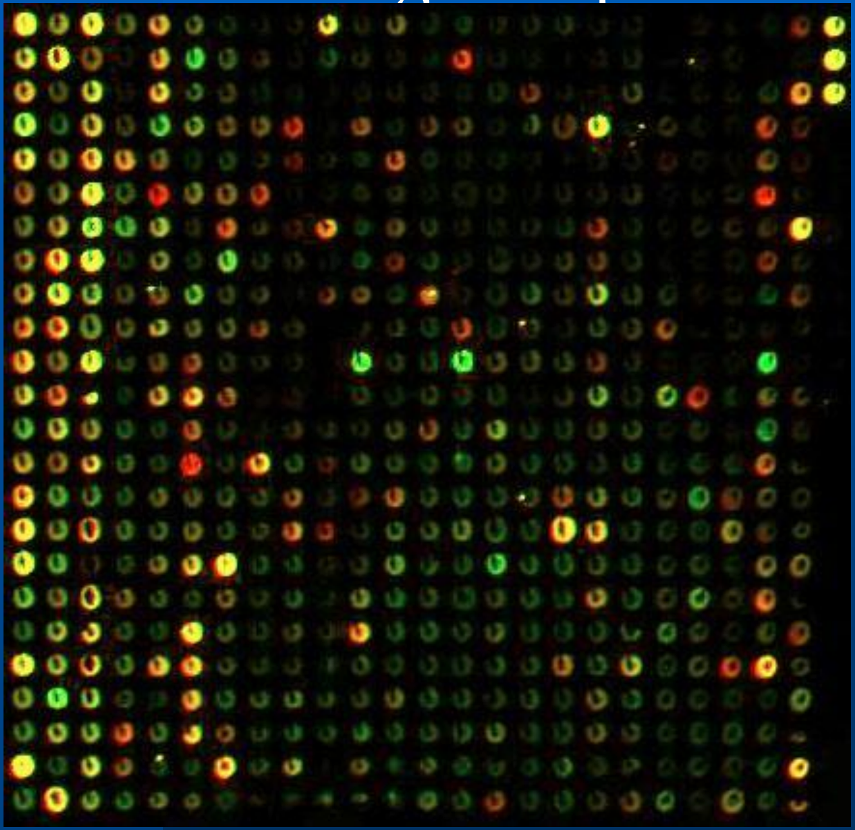
Cancer Class Discovery and Prediction by Gene Expression Monitoring



- Gene expression data for cancer prediction
  - Training data: 38 leukemia samples (27 ALL , 11 AML)
  - Test data: 34 leukemia samples (20 ALL , 14 AML)
  - Datasets contain measurements corresponding to ALL and AML samples from Bone Marrow and Peripheral Blood.
- Graphical models used:
  - Bayesian networks
  - Non-negative matrix factorization
  - Generative topographic mapping

Applications of GTM for Bio Data Mining (1)

- DNA microarray data provides the whole genomic view in a single chip.



(Figure from http://www.gene-chips.com/sample1.html)

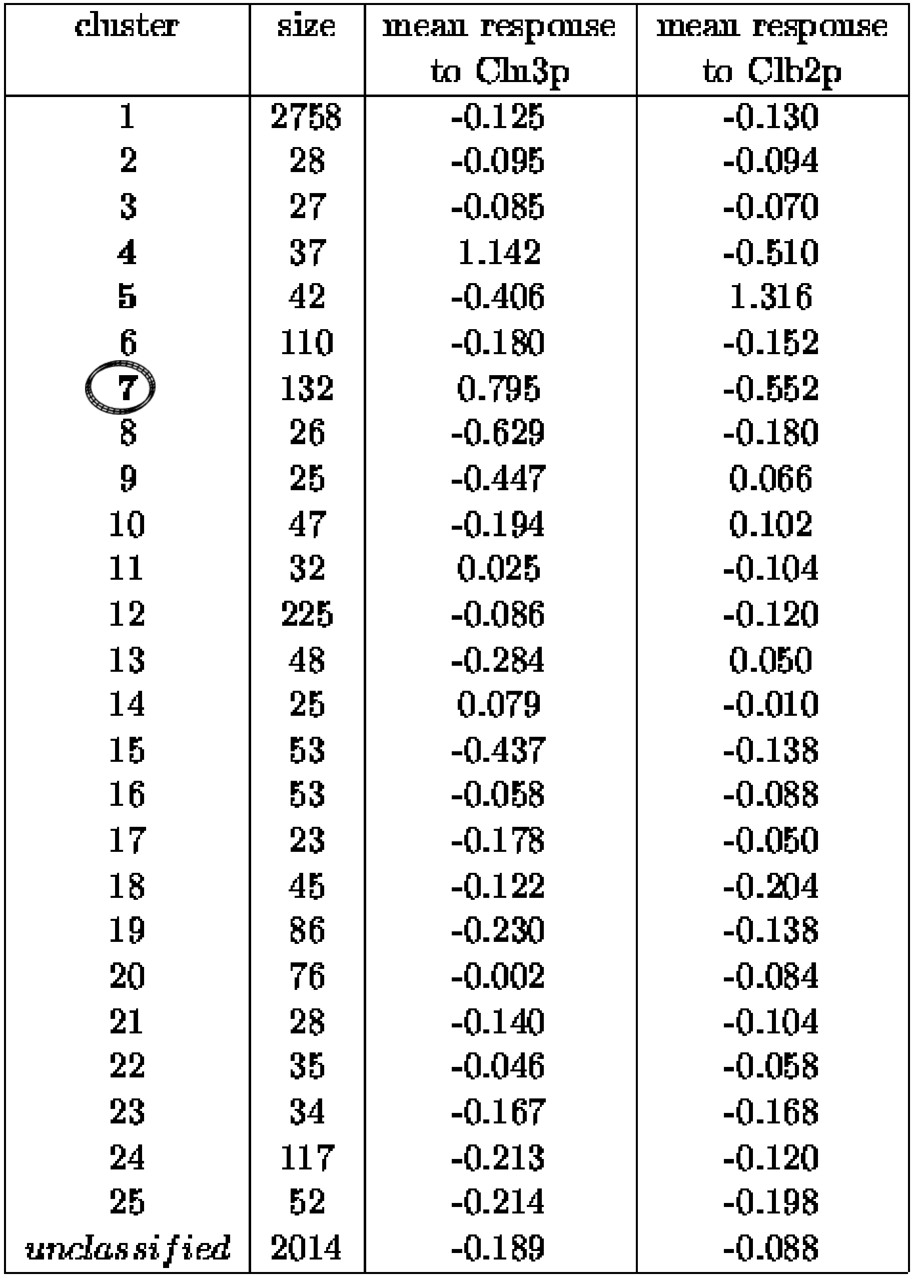
- The intensity and color of each spot encode information on a specific gene from the tested sample.

- The microarray technology is having a significant impact on genomics study, especially on drug discovery and toxicological research.

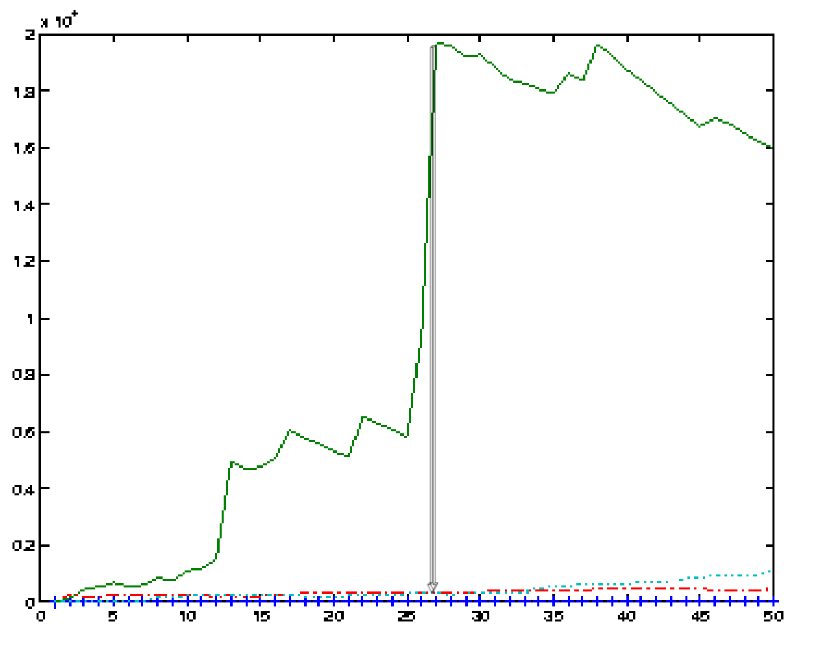
Applications of GTM for Bio Data Mining (2)

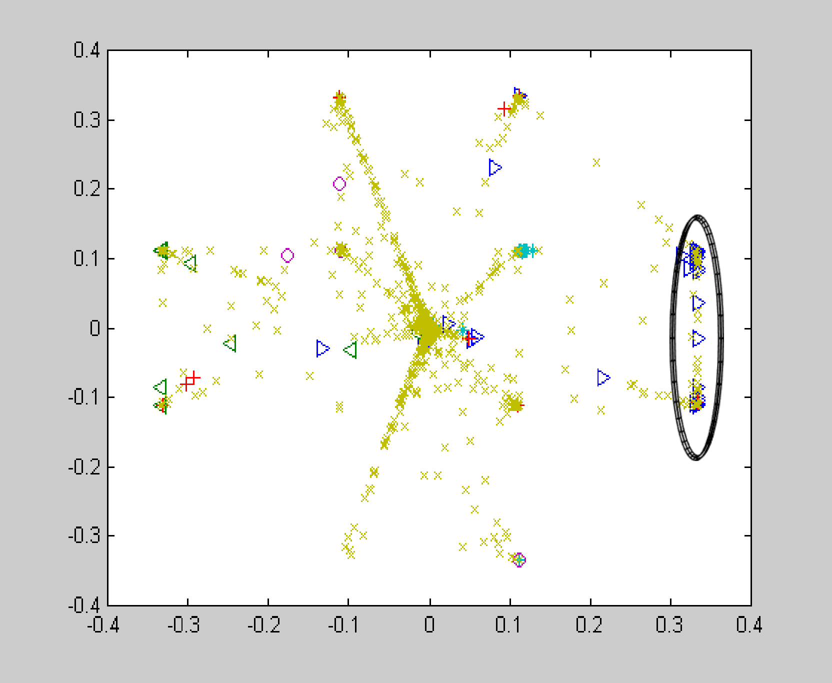
- Select cell cycle-regulated genes out of 6179 yeast genes. (cell cycle-regulated : transcript levels vary periodically within a cell cycle)
- There are 104 known cell cycle-regulated genes of 6 clusters
  - S/G2 phase : 9 (train:5 / test:2)
  - S phase : 8 (Histones) (train:5 / test:3)
  - M/G1 boundary (SWI5 or ECB (MCM1) or STE12/MCM1 dependent) : 19 (train:13 / test:6)
  - G2/M phase: 15 (train: 10 / test:5)
  - Late G1, SCB regulated : 14 (train: 9 / test:5)
  - Late G1, MCB regulated : 39 (train: 25 / test:12)

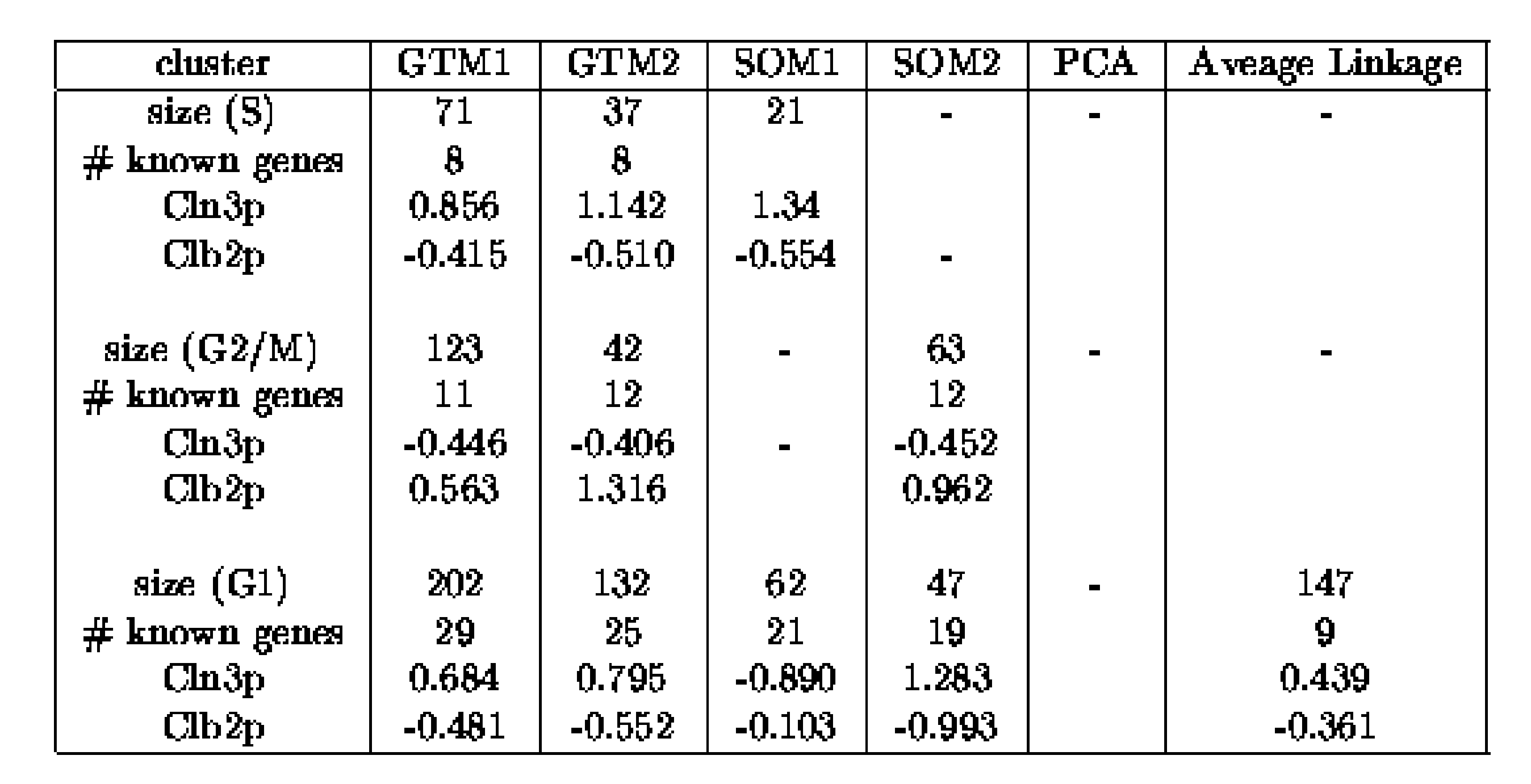
(M-G1-S-G2-M)



cluster	size	mean response to Clu3p	mean response to Clb2p
1	2758	-0.125	-0.130
2	28	-0.095	-0.094
3	27	-0.085	-0.070
4	37	1.142	-0.510
5	42	-0.406	1.316
6	110	-0.180	-0.152
7	132	0.795	-0.552
8	26	-0.629	-0.180
9	25	-0.447	0.066
10	47	-0.194	0.102
11	32	0.025	-0.104
12	225	-0.086	-0.120
13	48	-0.284	0.050
14	25	0.079	-0.010
15	53	-0.437	-0.138
16	53	-0.058	-0.088
17	23	-0.178	-0.050
18	45	-0.122	-0.204
19	86	-0.230	-0.138
20	76	-0.002	-0.084
21	28	-0.140	-0.104
22	35	-0.046	-0.058
23	34	-0.167	-0.168
24	117	-0.213	-0.120
25	52	-0.214	-0.198
unclassified	2014	-0.189	-0.088

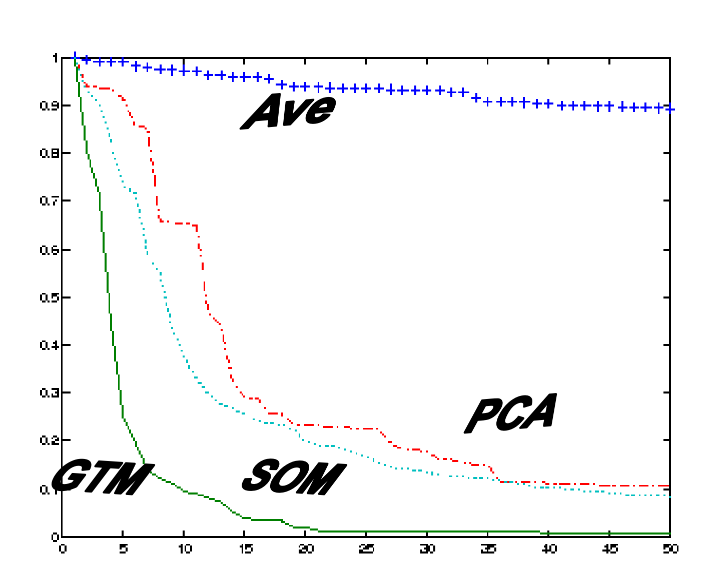






cluster	GTM1	GTM2	SOM1	SOM2	PCA	Aveage Linkage
size (S)	71	37	21	-	-	-
# known genes	8	8				
Cln3p	0.856	1.142	1.34			
Clb2p	-0.415	-0.510	-0.554	-		
size (G2/M)	123	42	-	63	-	-
# known genes	11	12		12		
Cln3p	-0.446	-0.406	-	-0.452		
Clb2p	0.563	1.316		0.962		
size (G1)	202	132	62	47	-	147
# known genes	29	25	21	19		9
Cln3p	0.684	0.795	-0.890	1.283		0.439
Clb2p	-0.481	-0.552	-0.103	-0.993		-0.361

[Clusters identified by various methods]



[The comparison of entropies for each method]

Summary and Discussion

- Challenges of Artificial Intelligence and Machine Learning Applied to Biosciences
  - Huge data size
  - Noise and data sparseness
  - Unlabeled and imbalanced data
  - Dynamic Nature of DNA Microarray Data
- Further study for DNA Microarray Data by GTM
  - Modeling of dynamic nature
  - Active data selections
  - Proper measure of clustering ability

References

- [Bishop C.M., Svensen M. and Wiliams C.K.I. (1988)]. GTM: The Generative Topographic Mapping, *Neural Computation*, 10(1).
- [Kohonen T. (1990)]. The Self-organizing Map. *Proceedings of the IEEE*, 78(9): 1464-1480.
- [P.T. Spellman, Gavin Sherlock, M.Q. Zhang, V.R. Iyer, Kirk Anders, M.B. Eisen, P.O. Brown, David Botstein, and Bruce Futcher. (1998)]. Comprehensive Identification of Cell Cycle-regulated Genes of the Yeast Saccharomyces cerevisiae, *Molecular Biology of the Cell*, Vol. 9. 3273-3297.
- [Pablo Tamayo, Donna Slonim, Jill Mesirov, Qing Zhu, Sutisak Kitareewan, Ethan Dmitrovsky, Eric S. Lander, and Todd R. Golub (1999)] Interpreting patterns of gene expression with self-organizing maps: Methods and application to hematopoietic differentiation. *Proc. Natl. Acad. Sci. USA Vol. 96, Issue 6, 2907-2912*
- [Cho, R. J., *et al*. (1998)]. A genome-wide transcriptional analysis of the mitotic cell cycle. *Mol. Cell 2, 65-7[3*.
- [W.L. Buntine (1994)]. Operations for learning with graphical models. *Journal of Artificial Intelligence Research* ,2, pp. 159-225.